Final Degree Project (TFG)

Degree in Industrial Technologies Engineering

Numerical simulations of inhomogeneous MHD flow in magnetically confined plasma

Bachelor thesis

Author: Francesc Xavier Ruché Alvarez
Supervisor: Shimpei Futatani

Date: January 2021

Technical School of
Industrial Engineering of Barcelona

Abstract

Plasma instabilities which are driven by spatially inhomogeneity variables are an unresolved problem yet for plasma physics and engineering applications. A useful approach to understand the global dynamics of plasma is the magnetohydrodynamics (MHD) theory. The inhomogeneous plasma parameters and magnetic fields are universal in magnetically confined plasma.

The project has been carried out using OpenFOAM which is an open source, free-licensed software, implementing the spatially inhomogeneous plasma transport coefficients and the magnetic fields in magnetically confined plasma. The analysis of the inhomogeneous MHD dynamics has been performed to understand the stability limit compared between uniform and non-uniform visco-resistive fields.

The results obtained by performing simulations on the selected domains prove the modified solver performs correctly on the pursued objective of accounting for the non-uniform resistivity and viscosity on the plasma. Further work needs to be done to prove the solver capabilities regarding the solution on non-uniform fields.

Acknowledgements

I would like to express my deepest appreciation to my supervisor during this work, Dr. Shimpei Futatani firstly for supervising during a work in such a fascinating topic for my undergraduate thesis and also for the amount of patience and dedication he has put into helping me develop this work.

I would like to extend my sincere thanks to Daniel Suarez Cambra, who has given his counsel throughout my thesis in order to solve innumerable problems regarding the OpenFOAM package.

I appreciate the assistance of both Joaquim Serra and Victor Jacoberger as their parallel work in OpenFOAM has been of great help for the current work.

The author thankfully acknowledges the computer resources at Marconi-Fusion and the technical support provided by the High Performance Computer at the CINECA headquarters in Bologna (Italy) for its provision of supercomputing resources.

Summary

Table of figures

Glossary

List of symbols

B	magnetic field
E	electrical field
H	magnetic field strength
M	magnetization vector
P	polarization density field
J	current density field
D	electric displacement field
u	velocity field
t	time
q	electrical charge
f	force
σ	electrical conductivity
ρ	density
p	plasma pressure
η	resistivity
ρ_c	charge density
ε	electrical permitivity
ε_0	electrical permitivity in vacuumm
μ	magnetic permeability
μ_0	magnetic permeability in vacuum
λ_D	Debye length

Introduction

OpenFOAM is an Open Source computational fluid dynamics C++ library that allows the creation of applications to solve fluid dynamics cases [1]. OpenFOAM provides the user a wide range of physics solvers for different fluid dynamics situations making it a very versatile software tool. The usage of this software tool in the academic environment is strongly encouraged by its Open Source nature, allowing researchers to view and modify the code to provide new solutions and applications, as well as its wide user community.

Among the different solver options provided with standard OpenFOAM the mhdFoam solver found under the electromagnetic solver classification is especially indicated to solve incompressible conducting fluid equations. The solver has been demonstrated to yield physical accurate results by previous works providing benchmark cases on toroidal and cubical geometrical domains [2].

Nonetheless, the mhdFoam solver physics are not completely realistic as some processes and capabilities are not provided. Such missing features include the passive scalar model for temperature transfer and non-constant visco-resistive effects on the fluid.

Passive scalar transport of temperature in MHD plasmas is an important topic to be researched as an addition to mhdFoam to improve physical model completeness. The passive scalar transport equation serves as a physical transport method for temperature on the plasma without incurring in active scalar effects by not modifying the local plasma properties such as density or viscosity. This improvement on the mhdFoam solver is being carried by the fellow student of the UPC Victor Jacoberger under the supervision of Dr. Shimpei Futatani and in close collaboration with the present work. The passive scalar model is useful to understand the characteristics of turbulence and has been studied to characterize the turbulence intermittency in the plasma turbulence [3]–[5].

The active scalar which is self-consistently coupled with the background turbulence is a more advanced transport model than passive scalar transport which remains as a further addition to the mhdFoam to improve the solver's physical completeness. The active scalar model is a more realistic transport model and the impurity transport on plasma modeled with active scalar transport has been studied in several scientific works [6], [7]. The effect of the active scalar transport has been proved to modify plasma characteristics [8] and therefore active scalar transport is considered an important future improvement for the solver.

The visco-resistive effect is present in the current mhdFoam solver in a very simplistic approach which does not take into consideration changes in the viscosity and resistivity of the plasma locally and solves for an ideal plasma with constant viscosity and resistivity. The

addition of non-uniform resistivity and viscosity solving capabilities to the solver is thought to be of great interest in improving the solver physical behaviour. The present work aims at providing a justified and physically coherent modification to the existing mhdFoam code to add non-uniform visco-resistive solving capability. It is worth mentioning that visco-resistive MHD physics have been studied in magnetically confined plasma using penalization methods in previous published works [9].

Objectives

The main objective of the present work is to expose the code structure and numerical methodology of the magnetohydrodynamics solver provided with OpenFOAM and explain and justify the adequate modifications proposed in order to extend its functionality to account for non-uniform visco-resistive plasma fluids.

The code will be explained and justified and several tests will be conducted in order to prove if the code is well suited for plasma simulation or if further research should be performed in order to achieve this goal.

Project's scope

Within the scope of the project the author will explain and detail the inner working mathematical mechanisms of the magnetohydrodynamics solver provided by the CFD package OpenFOAM.

The work will also provide a physics background in order for the reader to understand the mathematical development of the work carried out on the mhdFoam solver. The steps taken to modify the original mhdFoam solver will be detailed and justified in order for the reader to understand and be able to reproduce such modifications.

The project also comprises the creation of several test cases to benchmark the solver and determine if the proposed modifications enhance the solver by providing new solving capabilities for non-uniform visco-resistive fluids or on the other hand make the solver unstable or behave in a non-physical way.

The meshing procedure and the field generation is discussed for each test case in order to justify how the tests were performed and to what extent should they be taken as a proof that the solver does indeed work as planned.

Lastly the results of the simulated cases will be discussed in order to gain an understanding of how the solver has performed and if further modifications should be done to achieve completeness regarding non-uniform viscosity and resistivity.

Chapter 1: Fusion power introduction

1.1 Fusion technology

As the human population increases during the following generations a challenge will be faced regarding the energy need. Due to the growth and the advance of technology and living quality of the population, the current energy sources (mainly fossil fuels) will become scarce and insufficient to meet the energy demand. Therefore, new energy sources need to be found in order to meet that demand.

An alternative source has been found in nuclear fission, being a very cheap and efficient source of energy with no direct environmental footprint as any harmful material is exposed to the environment during normal operation or greenhouse effects gases. Nonetheless, nuclear fission is considered nowadays a dangerous technology if not operated correctly and with the highest safety standards as it has been demonstrated by nuclear disaster episodes such as the accidents of Fukushima and Chernobyl. Additionally, this technology poses a threat to the environmental safety as the sub products of its operation suffer from radioactive decay and therefore are considered dangerous for human and wildlife contact [10].

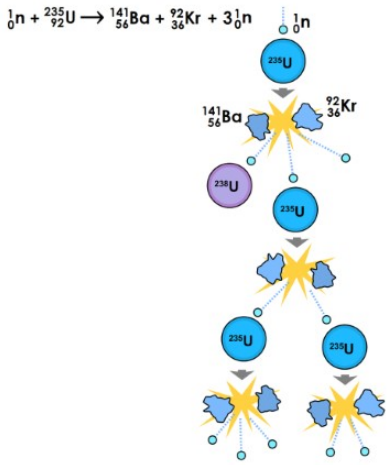

Figure 1: Fission chain reaction scheme of Uranium-235 [11].

Other alternative sources have been proposed, the so-called renewable energy sources, which comprise solar, marine, geothermal and wind power. These technologies rely on exploiting renewable resources which are replenished on a human timescale and therefore cannot be exhausted. Even though these sources may seem to be a solution to the energy demand, their technology is not advanced enough to provide a reliable and stable energy source due to poor efficiency factors and intermittent energy-production periods caused by the intermittent nature of some natural energy sources powering them (such as solar and

tidal power generation).

A candidate energy source has been found in nuclear fusion technology, which will be the focus of the present work. Nuclear fusion power generation relies on the opposite phenomena of the nuclear fission reaction, where a heavy atom's nuclei are bombarded with a neutron in order to split them into two new and lighter atoms (see Figure 1). The nuclear fission process yields energy by converting mass into energy as proposed in Einstein's special relativity theory according to the equality ($\Delta m\, c^2 = E$), where Δm is the difference between the mass of the initial atom and the combined mass of the two resulting atoms.

Three reactions are proposed by the scientific community to achieve a power-generation technology as they are considered the most energetically dense reactions (Equations 1-4). These reactions are the Deuterium-Deuterium reaction, the Deuterium-Helium 3 reaction, and the Deuterium-Tritium reaction [10]:

$$ {}^{2}_{1}H + {}^{2}_{1}H \longrightarrow {}^{3}_{2}He\ (0.82\ MeV) + n\ (2.45\ MeV), \tag{1} $$

$$ {}^{2}_{1}H + {}^{2}_{1}H \longrightarrow {}^{3}_{1}H\ (1.01\ MeV) + p\ (3.02\ MeV), \tag{2} $$

$$ {}^{2}_{1}H + {}^{3}_{2}He \longrightarrow {}^{4}_{2}He(3.6\ MeV) + p\ (14.7\ MeV), \tag{3} $$

$$ {}^{2}_{1}H + {}^{3}_{1}H \longrightarrow {}^{4}_{2}He(3.5\ MeV) + n(14.1\ MeV). \tag{4} $$

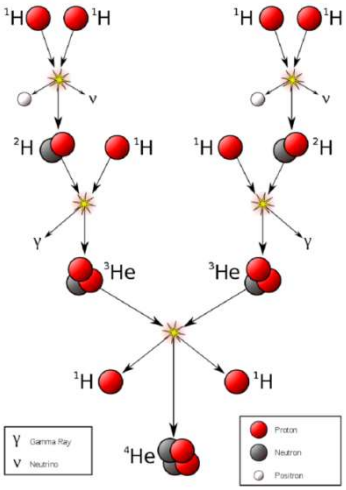

Figure 2: Deuterium-Deuterium fusion reaction scheme. Source: https://en.wikipedia.org/wiki/File:FusionintheSun.svg. Licensed under an Attribution 4.0 International (CC BY 4.0) license

From the three chosen nuclear fusion reactions, the most desirable is the Deuterium-Deuterium reaction (Eq. 1-2) as its fuel is easily obtained in great quantities from water which leads to a virtually unlimited energy source but nonetheless it is the most difficult reaction to initiate among the chosen reactions. The reaction has two branches with equal probability yielding Tritium and Helium 3 as well as protons and neutrons. Due to the generation of neutrons the reaction produces activation in the material's impurities surrounding the reactor chamber.

The reaction produces 0.82 and 1.01 MeV per nucleon respectively, which is acceptable within the nuclear power standards, but due to the difficulty to initiate the reaction, it was not chosen as the main focus of the current fusion research effort.

The Deuterium-Helium 3 reaction (Eq. 3) produces the impressive amount of energy of 3.66 MeV per nucleon which is high by nuclear standards. This reaction produces only charged products as sub products, which is considered a positive outcome as charged particles are easier to control through magnetic and electric fields from the engineering perspective. It also opens the possibility of converting energy directly to electricity by trapping the electrical charge of the products.

Nonetheless, the reaction was not chosen as the main reaction in research due to the difficulty to initiate it (less than in Deuterium-Deuterium reactions) and the helium 3 being a scarce resource.

The final reaction proposed which is the chosen one by the fusion research community is the Deuterium-Tritium reaction (Eq. 4). The reaction yields 3.52 MeV per nucleon, which is a high energy by nuclear standards, but it implies the usage of Tritium, which is a radioactive isotope of hydrogen with a short half-life (12.26 years) that makes it an extremely scarce resource.

The reaction was chosen as it is the easiest of them all to initiate and the scarce Tritium can be generated inside the reactor by reaction of the sub product neutrons with a lithium blanket (see Figure 3) on the reactor walls. The following reactions describe the Lithium breeding process:

$$ {}^{6}_{3}Li + n\,(slow) \longrightarrow {}^{4}_{2}He + {}^{3}_{1}H + 4.8\,MeV, \tag{5} $$

$$ {}^{7}_{3}Li + n\,(fast) \longrightarrow {}^{4}_{2}He + {}^{3}_{1}H + n - 2.5\,MeV. \tag{6} $$

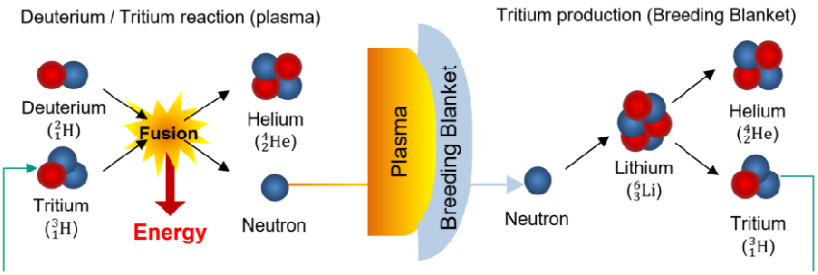

Figure 3: Schematic diagram of the tritium breeding inside a fusion reactor. [12].

The main issue regarding the nuclear fusion with whichever reaction is chosen is the temperature that needs to be achieved in order to force the nuclei of the fuel atoms together to fuse. This temperature needs to be high enough to overcome the electric repulsion from both fuel atoms, as both nuclei are positively charged. The temperature for the Deuterium-Tritium reaction is in the order of hundreds of millions of degrees Celsius.

At these kinetic energies the electrons are able to escape the electric attraction produced by the nucleus and therefore the fuel becomes ionized. The fuel substance becomes extremely hot and charged and effectively becomes plasma which needs to be confined and isolated from the reactor machine (as no material can withstand the temperatures achieved at the plasma), therefore a new kind of confinement technology must be used. This technology is magnetic confinement, which uses a combination of magnetic fields to contain the plasma suspended in the vacuum preventing it from touching the inner walls of the reactor and damaging them [10]. The main focus of this project is the study of the mechanics behind the plasma in order to simulate its behaviour more accurately.

Nuclear fusion as explained in this introductory chapter has the advantage over nuclear fission of being **clean** as in does not produce any harmful sub products, **cheap** and **long-lasting** as it requires hydrogen and lithium as fuel and these resources are very abundant and cheap to obtain.

Regarding the **safety**, nuclear fusion is much safer than fission due to its reaction needing energy feedback in order to continue, as opposed by nuclear fission which relies on chain reactions without feedback to operate making it an extremely dangerous system if left unattended. Also, the amount of fuel and therefore energy inside the reactor at a given time is extremely low compared to the fission reactors, this fact becomes a safety measure as in the event of a system failure, the reaction would stop rapidly due to a loss on the fuel supply system, feedback heating system or on the confinement system. The loss of control would lead to a halt of the power production and possibly damage in the reactor, but it wouldn't pose a danger to population and/or environment as in a fission reactor control loss.

Also the sub products of the fusion reaction pose no threat as they are not radioactive or chemically harmful. The only possible source of radioactive harm would come from the materials inside the reactor chamber due to activation of certain impurities due to neutron irradiation.

1.2 Fusion machines

One of the objectives of the MHD study and the one this thesis will explore is the usage of the MHD physics in the containment of fusion plasmas. As explained, the fuel of a fusion machine needs to be ionized and at great temperatures in order to have the sufficient kinetic energy to overcome the nuclear forces between nuclei to start a fusion reaction. In order to maintain the fuel in this state specific technologies have to be used to contain the plasma without physically contacting materials (as no known material can withstand the conditions imposed by the plasma due to its extremely high temperature).

In this section several of these technologies will be discussed in order to give a practical approach to the usage of MHD physics in fusion technology. Three of the most important fusion machines types will be briefly reviewed: Tokamaks, Stellarators and Magnetic mirror machines.

1.2.1 Tokamaks

Tokamak is the leading candidate fusion technology to achieve fusion reaction due to its excellent physics performance. The Tokamak technology consists of an axisymetric torus with a large toroidal magnetic field, a moderate plasma pressure and a relatively small toroidal current.

The tokamak topology consists of a toroidal inner space where hot plasma is confined by a system of magnetic fields developed by a group of coils located around the toroid (see Figure 4). A set of magnetic coils are present on Tokamak machines and allow the creation and control of several magnetic fields on the toroidal reactor vessel (see Figure 4). These coil set include the toroidal field coils, a set of vertical coil system used for force balancing inside the plasma chamber, a set of shaping coils which produce a non-circular cross-section to help improve MHD stability limits and help control plasma-wall impurities and an ohmic transformer which induces a toroidal plasma current required to heat the plasma [10].

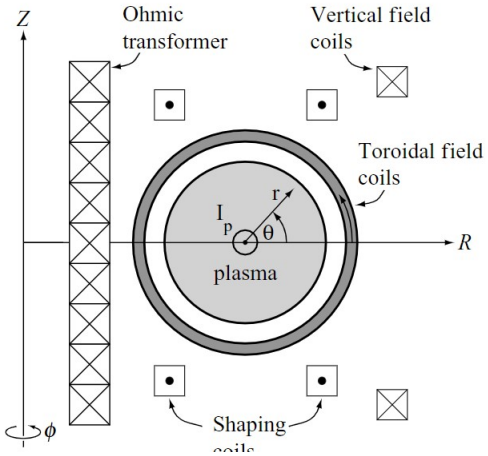

Figure 4: Tokamak simplified schematic. [10]

The tokamak operation starts by inducing a powerful toroidal magnetic field with its Toroidal field coils, then the fuel is injected into the toroidal chamber, often pre-ionized. Once the fuel is inside the reactor vessel the ohmic transformer is started and the fuel begins the heating process, during this stage neutral beams are also used to heat the plasma during the so called RAMP-UP PHASE of the reactor operation (see Figure 5). The ohmic heating transformer acts by the same principle as an induction transformer with the primary winding being at the centre of the ohmic transformer and the second winding being the plasma itself, the plasma. As the amount of current through the primary winding is limited by design and material factors and the current cannot be sustained during long periods of time, making Tokamak operation a pulsed process rather than a continuous operation process. In order to maintain the Tokamak operation longer other heating technologies ought to be used such as the neutral beam heating, which can sustain longer operation.

Once the ohmic heating of the plasma is achieved, the ohmic transformer operation is shut and the flat-top phase of the pulse is started. The interesting plasma physics behaviour and phenomena takes place during this flat-top phase as the characteristic equilibrium are achieved under this regime.

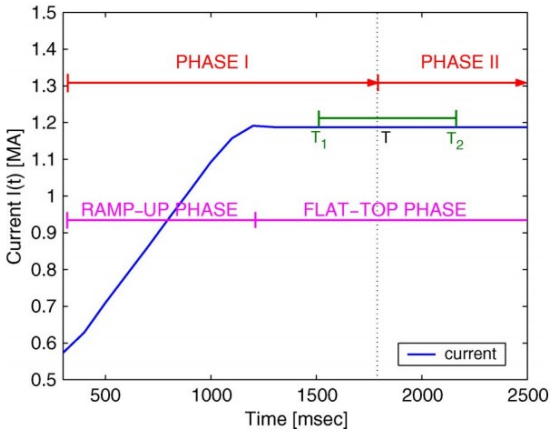

Figure 5: Tokamak plasma current evolution during ramp-up and flat-top operation of discharge. [13]

Tokamak reactors present an excellent physics performance due to their geometry and large magnetic toroidal field which leads to very stable plasma behaviour and high energy confinement times. Thanks to the good confinement capabilities of the Tokamak technology, high temperatures are achieved easily by using moderate amounts of energy as reactor input, minimizing the loss of energy on the vessel due to plasma leak.

The Tokamak technology demands very strong toroidal magnetic fields in order to achieve plasma confinement, and it becomes very technological demanding to operate for longer periods of time such high amounts of current through the toroidal coils. Also, as the reactor approaches steady-state regime (which is fundamental in obtaining a stable reaction which could be used as an energy source), ohmic heating has to be discarded as a heating technology due to its pulsed nature as explained before; therefore new and costly heating alternatives are being developed to replace ohmic heating technology in the quest for steady-state operation such as neutral beams and microwave heaters.

Tokamaks experience a kind of Neoclassical toroidal current production named Bootstrap current arises due to the presence of a pressure gradient associated with the existence of trapped particles. This Bootstrap current can provide up to the 95% of the toroidal current in the reactor, but the consensus among the fusion research community is that 75% of the toroidal current inside the reactor must be provided by Bootstrap current thus reducing the amount of external current induction to the 25%. Tokamaks designed to experience large bootstrap currents will need to stabilize the resistive wall mode [10].

Due to its excellent performance, numerous experimental facilities are operating Tokamak-like experiments, some of the most important ones include:

- JET: Named Joint European Torus is a fusion research facility located in Oxfordshire, United Kingdom which operates a magnetically confined plasma

physics experiment with Tokamak technology. It began its operation in 1983 in search for research and eventually reach the scientific breakeven, the point in which the reactor would produce enough energy to maintain a self-sustained reaction. The reaction inside JET uses Deuterium-Tritium as a fuel and reached a record for fusion energy production by producing 16MW of energy while being fed 24MW of thermal heating directly to the fuel; this record is the closest scientific community has been to a breakeven with a factor of $Q=0.67$. The ability to reach fusion breakeven was considered not possible for JET's tokamak design as several effects which impeded breakeven were not observed at the moment in previous fusion machines due to lower densities and pressures of plasma achieved.

The JET project resulted in a new tokamak design concept named "advanced Tokamak" which would improve Tokamak capabilities and hopefully allow reaching breakeven in future projects like ITER.

- ITER project: The ITER project, named International Thermonuclear Experiment Reactor is a nuclear research project aiming at designing, building and operating the first ever tokamak reactor to reach breakeven. The design specifications for the ITER project were to design a tokamak fusion machine capable of sustaining a fusion reactor yielding 500MW of heat power by only providing 50MW of heat power and being able to operate sustainably for around 20 minutes.

  The ITER project would demonstrate the feasibility of nuclear fusion technology by obtaining a $Q=10$ parameter meaning the reactor would yield 10 times the energy needed to sustain the reaction and therefore achieve breakeven, but the energy yielded at ITER would be vented as heat and not used as an energy source (as ITER is conceived as a research reactor facility). This event would trigger a new fusion technology project named DEMO with the aim of building the first commercially profitable fusion reactor and prove the economical feasibility of the technology.

1.2.2 Stellarators

Stellarators are the second major fusion reactor machines considered achieving controlled nuclear fusion reactions. In these machines, plasma is confined in a complex 3D geometry resulting as a helically symmetric system bent into a torus [10]. They are much more complex geometrical design (see Figure 6) and provide better stability on the plasma confinement than the Tokamak geometry.

The magnetic configuration of such devices is set by a large axisymetric toroidal field and a moderately sized helical field and a small axisymetric vertical field [10]. No ohmic heater is present in stellarator designs so the only current flowing in the plasma is the naturally occurring current which can be found on the plasma itself occurring just by means of natural

induction. In stellarators the large toroidal field provides a stabilization effect on the plasma itself, resulting in a stable plasma confinement without the need of a perfectly conducting wall like one would expect from Tokamak geometry.

Figure 6: Schematic diagram of the W7-X stellarator. [10].

Among the current stellarator devices being operated, three main projects can be described as the leading experiments on the stellarator study:

- The Large Helical Deice (LHD): located at the National Institute for Fusion Science in Japan, is a stellarator experiment built with a magnetic field configuration known as “heliotron”. The field configuration is created by two continuous interwined helical-toroidal coils and a set of axisymmetrical vertical coils that even though are considered a technology feat in fusion technology there’s a wide consensus in the opinion that extrapolating such coil design into a stellarator reactor does not provide the optimal design approach.

  LHD was designed to maximize MHD equilibrium and stability rather than single-particle confinement. When the experiment was designed, the MHD codes available for macroscopic simulation of plasma weren’t advanced enough to

- The Wendelstein 7-X: located at the Max Planck Institute in Germany is a stellarator research facility used to evaluate components for future fusion power plants. It’s the largest stellarator reactor in present use and was designed with the goal of achieving a continued plasma discharge during 30 minutes, thus demonstrating steady-state operation for stellarator fusion machines. By design the Wendelstein 7-X is intended to achieve a plasma density of 3e20 particles per cubic meter and a temperature of 120 million Kelvin.

1.2.3 Magnetic mirror machines

Magnetic mirror machines are one of the simplest approaches to magnetic confinement fusion machines. These machines consist in a strong magnetic field created by two isolated coils (see Figure 7). The magnetic field produced among the coils forces the charged particles into a spiral motion which confines the particles along the magnetic field lines. Once the particles reach the area close to the coils, the strong change in magnetic field strength forces the particles in the opposite direction, effectively producing a confinement.

Nonetheless, these machines experience particle losses at the coils which result quite substantial compared with other fusion machines technology.

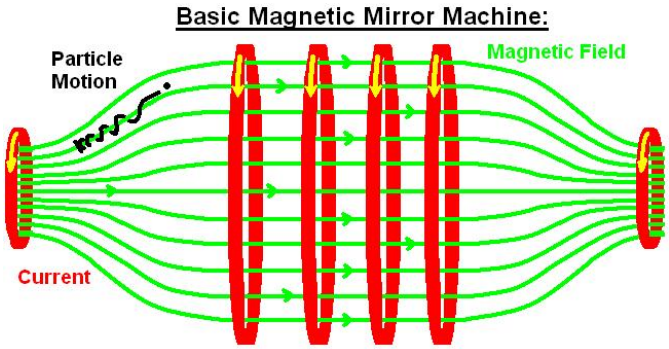

Figure 7 : Magnetic mirror machine scheme. Source: commons.wikimedia.org/wiki/File:Basic_Magnetic_Mirror.

The magnetic field in a mirror machine can be analytically derived from the Gauss's Law for magnetism keeping in mind the magnetic configuration in cylindrical coordinates and will be demonstrated in next chapters.

Some of the most important machines regarding this technology of magnetic confinement are:

- TMX: The Tandem Mirror Experiment, built at the Lawrence Livermore Laboratory, California was designed to perform plasma confinement experiments. Its configuration has 5 rings around the plasma and "baseball" shaped magnets at the end to keep plasma from escaping the machine. This design is considered quite complex and Tokamaks provide a much better alternative to the "baseball" magnets or the later installed "ying yang" magnets used to keep plasma in the machine.

- MFTF: The Mirror Fusion Test Facility was a magnetic mirror machine built at the Lawrence Livermore Laboratory, California, in 1986 using a tandem magnetic mirror design. It was never used due to economic issues and Tokamaks being a better alternative at the time of its construction. It would have contained plasma at an estimated 500 million Kelvin. The 350 ton magnetic mirrors in a "ying-yang"

configuration would provide a magnetic field inside a 58 m vacuum vessel containing the plasma.

- Gas Dynamics Trap: The Gas Dynamic Trap is a magnetic mirror built at the Budker Institute of Nuclear Physics, Russia, which consists in a cylinder of 7 meters long and 28 in diameter. The chamber contains a low magnetic field of 0.35 Tesla which increases to 15 Teslas at the ends of the tube due to the coils position. The plasma can be heated through a neutral beam of 5 MW. This machine has a continuous loss of plasma at its ends due to the mirror configuration and therefore fuel needs to be replenished continuously in order to maintain plasma density of up to $1\ e20$ ions/m3.

Chapter 2: Physics Introduction

2.1 Plasma

The main topic of this work is the understanding and simulation of plasma and therefore an understanding of the basics of plasma physics is mandatory.

Plasma, as described by [10] is:

"Fusion plasma is a fully ionized gas whose behaviour is dominated by long-range electric and magnetic fields (as opposed to short-range nearest-neighbor Coulomb collisions). A major consequence of this behaviour is that plasma is an exceptionally good conductor of electricity."

From this definition three criteria can be obtained in order to define plasma:

- It's an **ionized gas**: The most obvious criteria is the ionization of the constituent atoms of the plasma. The plasma is ionized due to its extremely high temperature, which translates into a high kinetic energy on its particles. As electrons gain kinetic energy, they become detached from the nucleus by having enough energy to escape the bound with the nucleus. A recombination process is also happening at the same time the atoms are being ionized, but by having a high enough temperature the ionization process becomes much faster than the recombination and the part of the plasma which is not ionized is so small it can be neglected.

- It presents a **collective behaviour**: The collective behaviour is defined by [10] by the phrase "its behaviour is dominated by long-range electric and magnetic fields" meaning the behaviour of the individual particles of the plasma is governed by the distant magnetic and electric fields rather than by the collisions with the near particles. As the density of the plasma increases the Columbian collisions among the particles also increase, making the collective behaviour of the plasma more difficult to obtain.

- The plasma is **quasi-neutral**: The plasma has overall neutrality, meaning it has no charge as a whole. Nonetheless, as the charges are free to move across regions of the plasma, their movement can raise areas of charge which can be rapidly neutralized. Even though the plasma is overall neutral these small areas of charge still produce a significant behaviour due to their electric field which cannot be neglected.

An effect which appears in plasma is the **Debye Shielding** consists on a sheath of charges that appears around a charged region of the plasma such as an electrode. The charged particles in the plasma gather round the electrode in order to maintain the quasi-neutrality of the area, but as they have a significant amount of kinetic energy some charges will escape the area of shielding and some will enter as well as recombination will occur. As a consequence of this behaviour the sheath has a certain thickness which is taken as a parameter of the plasma, the Debye length.

As temperature increases, the Debye length will also increase as the charged particles gain kinetic energy and the behaviour is reinforced.

Debye length becomes an important parameter describing the behaviour of the plasma to the point it can be used to determine if a substance can be treated a plasma or not, being L the characteristic length of the plasma.

$$\lambda_D = \left(\frac{\epsilon_0 KT}{e^2 n}\right)^2, \tag{7}$$

$$\lambda_D \ll L; \tag{8}$$

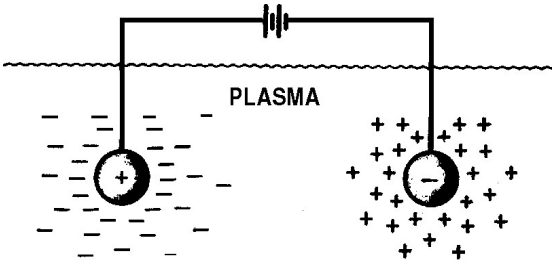

Figure 8: Debye shielding scheme. [14]

The Debye shielding is only valid if there are enough particles in the sheath, as if not enough particles are present around the charged region no effective shielding will appear and the quasi-neutrality condition of the plasma won't be fulfilled.

If the sheath is assumed to be a sphere, with radius λ_D, then the number of particles inside the sphere can be obtained as:

$$N_D = \frac{4}{3}\pi n {\lambda_D}^3, \tag{9}$$

$$N_D = 1380\frac{T^{3}/_{2}}{\sqrt{n}}; \tag{10}$$

And the shielding can be obtained if the number is much greater than one; as if there's less than one particle inside the Debye Sphere no particle will shield the particle generating the charged region [10].

$$N_D \gg 1; \tag{11}$$

If we imagine the Debye Shielding produced in plasma with two electrodes alternating its voltage, the charged particles will have to travel from one electrode to another in order to proper shield its electrode when they change their potential. So when the frequency of alternating is low enough the charged particles can travel to the other electrode when switching potential, but with higher frequencies they may not travel fast enough. The boundary of the existence of a Debye Shielding is denoted with the electron frequency:

$$\omega_{pe}{}^2 = \frac{nq_e^2}{m_e \varepsilon_0}, \tag{12}$$

And the Debye Shielding condition is fulfilled by the following equation with τ being the time between Columbian collisions.

$$\tau > \frac{1}{\omega_{pe}}. \tag{13}$$

2.2 Magnetohydrodynamics

Plasma is a gas with electrically charged particles as it is ionized. Therefore, the mechanics describing the behaviour of the plasma are those of a fluid coupled with the magnetic and electric behaviour. The theory describing the behaviour of plasmas is the MagnetoHydroDynamics (MHD). The particles resulting from the ionization are subject to the electrical and magnetic laws, called the Maxwell equations [15].

$$\nabla \times H = J + \frac{\partial D}{\partial t}, \tag{14}$$

$$\nabla \times E = -\frac{\partial B}{\partial t}, \tag{15}$$

$$\nabla \cdot B = 0, \tag{16}$$

$$\nabla \cdot E = \frac{\rho_c}{\varepsilon_0}. \tag{17}$$

By order of appearance:

The Ampère-Maxwell Law (Eq. 14) describes how a magnetic field appears and rotates around a flux of charges. Given a packet of charges which travels in a certain direction, a magnetic field will appear perpendicular to the direction of movement of the particle packet.

The Maxwell-Faraday equation (Eq. 15) describes how a changing magnetic field produces a changing electric field perpendicular to the magnetic field direction.

The Gauss' Law for Magnetism (Eq. 16) describes how the divergence of a magnetic field is always 0 and therefore no sources or sinks can be ever found on a magnetic field. This law is responsible for the non-existence of magnetic monopoles.

The Gauss Law (Eq. 17) describes how an electric field is produced by electrical charges. As the divergence of the field equals the charge density (divided by the permeability) the electric field lines can only sink or source at a spatial region with charge. Therefore, the concept of electrical charges and dipoles is defined by this equation.

There's also behaviour referent to every material's properties which should be taken into account. The following relations describe how magnetic and electric fields interact with a material regarding its material properties.

$$B = \mu H + \mu_0 M, \tag{18}$$

$$D = \varepsilon E + P; \tag{19}$$

The Ohm's Law is responsible for the relation among current and electrical field in a given space region and the law of conservation of charge defines how the current is sourced or sunk by the derivative of charge. Finally, the Lorentz force describes the interaction between charged moving particles and the magnetic and electrical fields.

$$J_c = \sigma(E + u \times B), \tag{20}$$

$$\nabla \cdot J = -\frac{\partial \rho_c}{\partial t}, \tag{21}$$

$$f = q(E + u \times B); \tag{22}$$

As we stated the plasma behaves as a fluid under the influence of the magnetic and electric laws of dynamics. Therefore, we should also take into account the laws which govern a fluid, which are the Navier-Stokes equation for compressible Newtonian fluids, the Reynolds Transport Theorem and the continuity equation (in order of appearance).

$$\rho \frac{\partial v}{\partial t} + \rho(u \cdot \nabla)u - \frac{1}{3}\mu_f \Delta u + \nabla p = f, \tag{23}$$

$$\frac{d}{dt}\int_{\Omega(t)} f \, dV = \int_{\Omega(t)} \frac{\partial f}{\partial t} \, dV + \int_{\partial\Omega(t)} (u^b \cdot n) \, dA, \tag{24}$$

$$\frac{\partial \rho}{\partial t}(x,t) + \nabla \cdot (\rho u)(xt) = 0. \tag{25}$$

By combining the equations which define the behaviour of fluids and electromagnetic a set of equations can be obtained which model the behaviour of plasma and are the basis of the MHD model. The equations which result from the combination of these two models are the so-called "ideal MHD Model Equations" which comprises the following seven expressions [15]:

$$\frac{\partial \rho}{\partial t} \nabla \cdot (\rho u) = 0, \tag{26}$$

$$\rho\left(\frac{\partial}{\partial t}+u\cdot\nabla\right)u=J\times B-\nabla p, \tag{27}$$

$$E+u\times B=0, \tag{28}$$

$$\frac{\partial B}{\partial t}=-\nabla\times E\,, \tag{29}$$

$$\mu_0 J=\nabla\times B, \tag{30}$$

$$\nabla\cdot B=0, \tag{31}$$

$$\frac{d}{dt}\left(\frac{p}{\rho^{\gamma}}\right)=0. \tag{32}$$

MHD has multiple applications not only in fusion research but also in earth studies and astrophysics, where plasma is present and its behaviour can be studied by MHD. Also, MHD is a powerful tool not only for plasma study but also to describe any conducting fluid under electromagnetic force influence.

2.3 Description of the physics of code improvement

As it is discussed in the previous section the ideal MHD model is derived from the Maxwell equations and the fluid equations by neglecting the resistivity in the Ohm's Law equation. Nevertheless, this assumption cannot be made when talking about resistive plasmas as the resistivity can play a non-negligible role; therefore a more general approach of the Ohm's Law needs to be made.

$$E+u\times B=\frac{1}{en}\left(J\times B-\nabla p_e\right)+\eta J; \tag{33}$$

The previous expression is known as the "generalized" Ohm's law. The terms $J\times B$ and ∇p_e terms correspond to the Hall term and electron diamagnetic term respectively. In certain situations these terms may play an important role but in MHD these two terms can be neglected.

$$E+u\times B=\eta J; \tag{34}$$

By neglecting the discussed terms we obtain a "resistive" Ohm's Law. To obtain the ideal MHD model the resistivity would be neglected due to its smallness but in the resistive model it is maintained because resistivity is the only dissipative process appearing in the momentum equation (the Hall and electron diamagnetic terms are non-dissipative) [15].

Replacing the resistive Ohm's Law in the ideal MHD model and equating we can obtain a model for a uniform resistivity and a model for a spatial-dependent resistivity. The development of both models is described below:

Uniform resistivity

Under the assumption of a non-ideally conducting plasma the electrical resistivity of the fluid (also named magnetic diffusivity in fusion technology) is non-null and therefore needs to be taken into consideration when deriving the magnetic model equations (the magnetic field time derivative in particular) by adding the eta parameter for electrical resistivity in the Ohm's law (η).

$$\begin{cases} E + u \times B = \eta J \\ \dfrac{\partial B}{\partial t} = -\nabla \times E \end{cases}, \tag{35}$$

By simple mathematical manipulation, the electrical field is incorporated to the magnetic time derivative. Taking into consideration the presented system of equations, the expression can be equated into its model form which is implemented in the mhdFoam solver.

$$\frac{\partial B}{\partial t} = -\nabla \times (\eta J - u \times B), \tag{36}$$

$$\frac{\partial B}{\partial t} = \nabla \times (u \times B) - \nabla \times (\eta(\nabla \times B)), \tag{37}$$

The uniform model for resistivity can be easily deducted from the resistive Ohm's Law and the Faraday's Law of Induction bearing in mind the following equalities and vector identities.

$$J = \nabla \times B, \tag{38}$$

$$\nabla \cdot B = 0, \tag{39}$$

$$\nabla \times (\varphi(\nabla \times A)) = \varphi \nabla \times (\nabla \times A) + \nabla \varphi \times (\nabla \times A), \tag{40}$$

$$\nabla \times (\nabla \times A) = \big(\nabla(\nabla \cdot A)\big) - \nabla^2 \mathrm{A}. \tag{41}$$

The field operator identity is then used to equate the term $\nabla \times (\eta(\nabla \times B))$ and decompose it into the following term:

$$\nabla \times (\eta(\nabla \times B)) = \eta \nabla \times (\nabla \times B) + \nabla \eta \times (\nabla \times B) = \eta \nabla \times (\nabla \times B) + 0, \tag{42}$$

$$\eta \nabla \times (\nabla \times B) = \eta\big(\nabla(\nabla \cdot B)\big) - \eta \nabla^2 \mathrm{B} = 0 - \eta \nabla^2 \mathrm{B}, \tag{43}$$

$$\frac{\partial \mathrm{B}}{\partial \mathrm{t}} - \eta \nabla^2 \mathrm{B} - \nabla \times (\mathrm{u} \times \mathrm{B}) = 0. \tag{44}$$

With the latter being the actual model equation implemented on the mhdFoam solver as it will be shown on the next chapters. As stated this model equation is only valid for uniform resistivity fields as the gradient of eta is considered zero for a uniform resistivity field ($\nabla \eta = 0$).

Spatial dependent resistivity

In order to accomplish the objective of the present work the model equation needs to be reformulated by accounting for a non-uniform resistivity field (for a non-uniform viscosity field the implementation on mhdFoam is already valid). The start point of the new model equation is the exact same as the model equation for the non-spatial dependent resistivity field. The Ohm's Law is formulated for a non-'ideal MHD' plasma accounting for a resistivity magnitude (eta).

$$\begin{cases} E + u \times B = \eta J \\ \dfrac{\partial B}{\partial t} = -\nabla \times E \end{cases}, \tag{45}$$

The procedure to formulate the model equation is the same as exposed previously in the uniform resistivity model equation. The resistivity has now to be considered non-uniform and therefore its gradient cannot be neglected during mathematical manipulation as done previously ($\nabla \eta \neq 0$).

$$\frac{\partial B}{\partial t} = -\nabla \times (\eta J - u \times B), \tag{46}$$

$$\frac{\partial B}{\partial t} = \nabla \times (u \times B) - \nabla \times (\eta (\nabla \times B)), \tag{47}$$

During this mathematical manipulation step, the term $\nabla \eta \times (\nabla \times B)$ is not neglected and thus appears in the final model equations;

$$\nabla \times (\eta (\nabla \times B)) = \eta \nabla \times (\nabla \times B) + \nabla \eta \times (\nabla \times B), \tag{48}$$

$$\eta \nabla \times (\nabla \times B) = \eta \big(\nabla (\nabla \cdot B)\big) - \eta \nabla^2 \mathrm{B} = 0 - \eta \nabla^2 \mathrm{B}, \tag{49}$$

$$\frac{\partial \mathrm{B}}{\partial \mathrm{t}} - \eta \nabla^2 \mathrm{B} - \nabla \times (\mathrm{u} \times \mathrm{B}) + \Big(\nabla(\eta) \times \big(\nabla \times (\mathrm{B})\big)\Big) = 0. \tag{50}$$

As it can be seen in the equation model, the term $-\Big(\nabla(\eta) \times \big(\nabla \times (\mathrm{B})\big)\Big)$ has been introduced

to the model equation to account for non-uniform resistivity on the plasma. The introduction of this modification into the solver is the main objective of this bachelor thesis. The present work aims at integrating the necessary physics equations and concepts into OpenFOAM through the modification and programming of the MHD solver package to include the capability of solving MHD cases where the resistivity and viscosity of the plasma is not set or constant but changes accordingly to a given profile. Also, as the viscosity and resistivity are related and dependent on the temperature, the solver will be modified to account for that behaviour. Therefore, the main objective of the thesis will be the addition of a spatial dependent resistivity model to the mhdFoam solver.

Chapter 3: OpenFOAM

In this chapter the main features of the OpenFOAM code are detailed regarding the solving methodology of the numerical cases. An introduction on the main workings of OpenFOAM is given as well as a detailed explanation of the solver's of interest code (the mhdFoam solver).

The mathematical insights of the code are discussed and explained as well as the algorithms used in the solving procedure (The PISO algorithm and its usage as the B-PISO variant).

By reading this chapter the reader is supposed to gain a reasonable understanding of how the solver works and how its code is structured in order to make the appropriate changes proposed in the next chapter.

3.1 Introduction to OpenFOAM

OpenFOAM (Open Source Field Operation and Manipulation) is a C++ library used to create executables known as an application. The executables are created by compiling the pre-existing or modified code provided with the package and are separated into two possible categories: *solvers* and *utilities* [1].

The *solver* applications contain the mathematical, physical and numerical methods to perform the simulation on a supplied case for a specific problem in continuum mechanics. OpenFOAM offers a wide range of numerical solvers for different physics problems.

Utilities on the other hand are applications designed to perform tasks which require data-manipulation such as generating physical fields, performing post and preprocessing of the obtained data during the simulation.

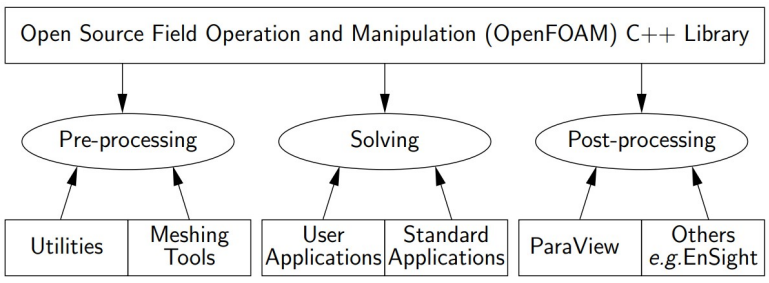

Figure 9: OpenFOAM structure [1]

3.1.1 OpenFOAM case structure

Cases designed to be solved by using OpenFOAM are contained in folders with a predefined

file structure. As the user should be familiar with the structure of the case in order to perform simulations a brief description of how OpenFOAM structures these folders is given.

The case folder contains at least 3 subfolders: 0, constant and system.

The 0 subfolder is a time folder. This kind of folders store the information for each time step of the solution and during the simulation procedure these folders will automatically created by OpenFOAM according to the runtime time of the time step. Inside the user will find the files containing a detailed description of each field necessary and involved in the solution process. Different solvers may produce and need different fields. The first time folder labelled 0 must contain the fields for the initial configuration of the case. In this work the created fields detailed during the Pre-processing chapters will be saved into this specific folder to impose the initial conditions on the cases.

The system subfolder contains a set of OpenFOAM dictionaries to allow for configuration of parameters such as the time step writing frequency, the solver control parameters, numerical solver schemes and several other configuration properties. Among the files contained in this folder, the controlDict and fvSchemes are the most important ones to tweak with the solver capabilities. If the mesh needs to be decomposed in order to be run in a multi-CPU system with parallelization, an additional decomposeParDict file needs to be created and configured.

The constant subfolder contains the information about the mesh in polyMesh format, by containing the different mesh patches, cell nodes and faces. It also contains the transportProperties dictionary which is used to configure constants into the solver operation such as viscosity or resistivity constants in the original mhdFoam solver.

3.2 Numerics behind OpenFOAM

Before diving into the OpenFOAM solver codes it is mandatory to explain the way in which the package is designed to mathematically deal with control volumes, being these the cells which form the mesh in which the fluid is simulated. The method used by OpenFOAM code is the Finite Volume Method which will be briefly detailed here.

In numerical CFD simulation and analysis the geometrical domain in which the fluid is simulated is divided into control volumes, being these the cells which compose the mesh. These control volumes can be of any size and shape as long as they're convex elements with planar faces. The control volumes can be internal if they're enclosed in other control volumes or external if they're adjacent to a geometrical domain's surface.

The control volume has a centroid point $\boldsymbol{P}$ around which the control volume $\boldsymbol{V_p}$ is constructed. A possible neighbor control volume $\boldsymbol{V_n}$ with a centroid point $\boldsymbol{N}$ can be placed alongside the

control volume V_p with which it will share a common face (see Figure 10).

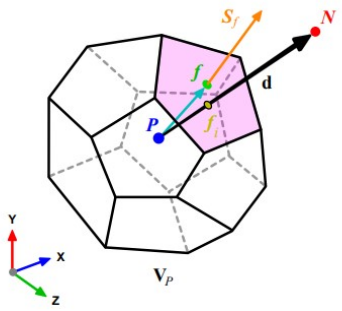

Figure 10: Example of a finite volume domain [16]

In the given notation, **f** denotes the centre of the face and $\mathbf{S_f}$ is the face vector, which is normal to the face f of the cell and has a magnitude which equals the face area. The point $\mathbf{f_i}$ denotes the intersection point between the **d** vector and the face f.

Given this notation for the cells on the computational meshes OpenFOAM stores the value for each field as the value at the centre of the centroid of the cell's volume. A scalar ϕ value for a field in the mesh is determined as:

$$\phi_P = \bar{\phi} = \frac{1}{V_P} \int_{V_P} \phi(x)\, dV. \tag{51}$$

A general transport equation for the scalar ϕ can be denoted as by means of the Reynolds transport theorem as:

$$\int_V \frac{\partial \rho \phi}{\partial t} dV + \int_V \nabla \cdot (\rho u \phi)\, dV - \int_V \nabla \cdot \left(\rho \Gamma_\phi \nabla \phi\right) dV = \int_V S_\phi\, (\phi) dV. \tag{52}$$

with the first integral term (from left to right) being the temporal derivative of the scalar ϕ, the second term being the convective term which is associated to the transport of the scalar by means of advection, the third term being the diffusion term responsible for the transport of ϕ due to differences of ϕ in the field and the latter being the source term responsible for the source or sink of the scalar in the field by means of external generation [16].

This transport theorem is fundamental in fluid dynamics as it represents the main phenomena driving scalar transport. In order to implement the Reynolds Transport Theorem by using the Finite Volume Method, the Gauss or Divergence Theorem

$$\int_V \nabla \cdot a\, dV = \oint_{\partial V} dS \cdot a. \tag{53}$$

with ∂V being a closed surface bounding the control volume V_p , and dS being an infinitesimal surface element with the unity vector $\hat{n}$ (The control volumes and infinitesimal surfaces are associated with the cell's volume and its faces on the mesh). With the Gauss Theorem the Reynolds Transport Theorem can be rewritten in terms of the bounding surface in order to be further discretized.

$$\frac{\partial}{\partial t}\int_{V_p}(\rho\phi)\,dV + \oint_{\partial V_p} dS\cdot(\rho u\phi) - \oint_{\partial V_p} dS\cdot\left(\rho\Gamma_\phi\nabla\phi\right) = \int_{V_p} \mathrm{S}_\phi\,(\phi)dV. \tag{54}$$

The Transport Theorem consists now (from left to right) of the same temporal derivative, the convective term is now equated as a convective flux, the diffusive term is also rewritten as a diffusive flux term and lastly the source term is left unchanged. The flux terms on the faces of each cell are obtained by equating the sum of the fluxes on every cell's face to the surface integral denoted above. For the diffusive flux the flux through each surface is:

$$\oint_{\partial V_p} dS\cdot\left(\rho\Gamma_\phi\nabla\phi\right) = \sum_f \int_f dS\cdot\left(\rho\Gamma_\phi\nabla\phi\right)_f \approx \sum_f S_f\cdot\left(\rho\Gamma_\phi\nabla\phi\right)_f. \tag{55}$$

And for the convective term the procedure can be repeated as

$$\oint_{\partial V_p} dS\cdot(\rho u\phi) = \sum_f \int_f dS\cdot(\rho u\phi)_f \approx \sum_f S_f\cdot(\rho u\phi)_f. \tag{56}$$

The source term can also be approximated by a constant generation term S_c on the cell's volume and a variable generation term S_p influenced by the scalar ϕ_P

$$\int_{V_P} S_\phi\,(\phi)\,dV = S_c V_P + S_p V_P\,\phi_P m \tag{57}$$

The Transport equation is then obtained in the form:

$$\int_{V_p}\frac{\partial}{\partial t}(\rho\phi)\,dV + \sum_f S_f\cdot(\rho u\phi)_f - \sum_f S_f\cdot\left(\rho\Gamma_\phi\nabla\phi\right)_f = \left(S_c V_P + S_p V_P\,\phi_P\right). \tag{58}$$

Where the values appearing in the convective and diffusive terms are the values of the scalar and its properties computed at the cell's face from the value stored by OpenFOAM at the cell's centroid. To obtain the scalar at the face some kind of interpolation method ought to be used to take into consideration the two control volume's (the two which faces are adjacent) scalar value at their centroid.

The interpolation method used to calculate the values at the faces can be selected among many options, such as linear interpolation, upwind differencing and second order upwind differencing among others.

3.3 mhdFoam

3.3.1 Mathematical considerations

The solver for magnetohydrodynamics provided in the OpenFOAM package is named mhdFoam and is based on the B-formulation of the full set of magnetohydrodynamics equations [17].A bi-directional coupling between the magnetic and velocity fields is considered and the solenoidal nature of the magnetic field is determined as a key point in the solving process. The non-zero magnetic divergence can cause solution errors along the simulation procedure or even make the system unstable and unsolvable.

Several strategies have been proposed over the years to enforce the soleniodal nature of the magnetic field on MHD simulations but the one chosen by OpenFOAM programmers to solve MHD cases is the projection method proposed by [18].

The projection method proposes a correction to the magnetic field after the time step is completed by some arbitrary numerical scheme. The main idea is that the magnetic field obtained in a time step can be projected into a divergence-free magnetic field for the same time step.

The mathematical description of the procedure to execute the projection method is obtained from the original work [18] and explained in [17] as:

"*The method is based on the decomposition of a vector field into the sum of a curl and a gradient:*

$$B^* = \nabla \times A + \nabla\psi. \tag{59}$$

Where B is the predicted magnetic field, A is the vector potential and* ψ *is a scalar function. Taking the divergence on both sides, a Poisson equation for* ψ *is obtained:*

$$\nabla^2\psi = \nabla \cdot B^*. \tag{60}$$

The predicted magnetic field is then corrected by:

$$B = B^* - \nabla\psi. \tag{61}$$

To ensure that the obtained B is solenoidal, the Laplace operator in the ψ *Poisson equation*

must be evaluated in two steps, that is as a divergence of the gradient. Thus the divergence operator used in the Laplace operator must be the same as the one used for calculating ($\nabla \cdot B^*$) and the gradient operator used in the Laplace operator must be equal to the one used for calculating the gradient of phi. The correction on the magnetic field does not affect the current density as the additional term adds a gradient term and leaves the rotational value of the field untouched.

$$J = \nabla \times B = \nabla \times B^*. \tag{62}$$

With this method it is important to apply the boundary conditions to the corrected magnetic field and to choose good boundary conditions for the Poisson equation."

The algorithm included in mhdFoam is called B-PISO algorithm and it will be explained in the following section.

The mhdFoam solver executes a PISO iterative pressure-velocity coupling algorithm on the momentum equations to solve for an incompressible fluid and satisfy the continuity equation. Once the velocity field is calculated with the divergence error under the desired threshold the magnetic B-PISO loop starts and proceeds to solve the magnetic model equation and ensure a magnetic-divergence free magnetic field. If the PISO algorithm fails to find a solution to the U and B field which does not meet the divergence-free condition the solver crashes into an error due to a non-physical field phenomenon.

3.3.2 B-PISO algorithm

In this section, the B-PISO is explained as demonstrated by [17]. A general transport equation can be defined as:

$$A_v(v)v = b_v(v). \tag{63}$$

Defining D and G as the divergence and the gradient in discrete operator respectively, the set of governing equations can be written in discrete form for each node from the already denoted magnetic equations provided as the MHD model with uniform resistivity[17]:

$$D(v) = 0, \tag{64}$$

$$\frac{\partial v}{\partial t} + D(\phi_v v) - D\big(vG(v)\big) - D\left(\frac{\phi_B}{\rho\mu_m}B\right) + G\left(\frac{B^2}{2\rho\mu_m}\right) = -G\left(\frac{p}{\rho}\right), \tag{65}$$

$$\frac{\partial B}{\partial t} + D(\phi_v B) - D(\phi_B v) - D\left(\frac{1}{\eta}G(B)\right), \tag{66}$$

$$D(B) = 0. \tag{67}$$

Continuity and momentum coupling are solved following a pressure-based PISO-like algorithm in an iterative manner to provide a solution U field. The B-PISO algorithm is used to solve the magnetic transport equation described above and provide a magnetic field solution with non-divergence magnetic fields.

As stated in [17]:

"*In pressure-based algorithms, continuity equation is manipulated by means of momentum equation to obtain a pressure equation. Here momentum equation is used as a pre-conditioner for the velocity in order to improve the accuracy of the laplacian solver for the pressure equation. Once the new pressure is obtained, the velocity is corrected. Very schematically, and using the above mentioned matrix notation, the MHD algorithm used for the full set of equations. As can be seen, the Picard linearization method was already available in the official OpenFOAM version, except for the magnetic field correction step, that was implemented according to the Projection method from* [18].

Another improvement with respect to the original algorithm has been the splitting of the magnetic field in Bo, the externally applied magnetic field, and b, the induced one. This allows the user to impose a fixed external magnetic field (that do not need to be constant in time and/or in space), which is the common practice in fusion technology applications. Thus, new boundary conditions can be used (equations 5.5 and 5.6, being n the surface normal unit vector), where the user needs to define the initial map for Bo and b, and the B map is created as the sum of both, with Bo-type boundary conditions."

3.3.4 Pressure-velocity coupling

During the numerical solving process of a flow the convection of a scalar variable ϕ depends on the local velocity field direction and magnitude. In general, the velocity field for a fluid during its numerical solution is not known and rather emerges as part of the solution process. The transport equations for the velocity components can be obtained as the momentum equation for each coordinate direction (x,y,z). It is worth noting that each velocity component appears in each momentum equation and that the overall system must satisfy the continuity equation as well [19]. In the simple case of a two-dimensional steady flow, the equations can be written as:

$$\frac{\partial}{\partial x}(\rho u_x u_x) + \frac{\partial}{\partial y}(\rho u_x u_y) = \frac{\partial}{\partial x}\left(\mu \frac{\partial u_x}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu \frac{\partial u_x}{\partial y}\right) - \frac{\partial p}{\partial x} + S_{u_x}, \tag{68}$$

$$\frac{\partial}{\partial x}(\rho u_x u_y) + \frac{\partial}{\partial y}(\rho u_y u_y) = \frac{\partial}{\partial x}\left(\mu \frac{\partial u_y}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu \frac{\partial u_y}{\partial y}\right) - \frac{\partial p}{\partial y} + S_{u_y}, \tag{69}$$

$$\frac{\partial}{\partial x}(\rho u_x) + \frac{\partial}{\partial y}(\rho u_y) = 0; \tag{70}$$

The pressure term appears on both the momentum equations and the continuity equation and becomes the main momentum source term in most flows. These transport equations create additional solving issues as they contain non-linear quantities and are intricately coupled; each velocity component is present in the momentum equations and the continuity equation and the pressure magnitude is present in all the equations without any “pressure equation”.

Normally on numerical fluid simulations the pressure field for a flow is not known beforehand other than at initial time. An approach to obtaining the pressure field in order to solve the momentum and continuity equations could be to use the continuity equation as a transport equation for density and the energy equation as a temperature transport equation in order to compute the density and temperature locally and therefore calculate local pressure.

$$p = p(\rho, T). \tag{71}$$

This strategy is only valid for compressible fluid simulation as in incompressible fluid’s density becomes constant and thus it’s not linked to pressure transport. This issue introduces a constraint between pressure and velocity; if the pressure field is known the momentum equations can be solved and the resulting velocity field must satisfy the continuity equation.

To solve this problem regarding equation coupling and non-linearity the SIMPLE algorithm by Patankaar and Spalding is chosen to obtain correct velocity and pressure fields.

SIMPLE algorithm the convective fluxes through cell faces are obtained from a guessed velocity field. A guessed pressure field is applied to the momentum equations and the continuity equation, which now can be solved as pressure and velocity fields are known as a guess. The continuity equation provides a pressure correction field as the initial guess would have not satisfied the equation; the difference from the guessed pressure field and the field necessary to satisfy the continuity equation is applied to the pressure field and new corrected pressure and velocity fields are obtained. The SIMPLE algorithm relies on an iterative process to correct velocity and pressure fields from an initial guess and progressively improve the solution until convergence.

3.3.5 PISO algorithm

OpenFOAM's mhdFoam solver makes use of a pressure-velocity coupling algorithm as needed to solve for incompressible fluids. The developers of the solver used a PISO algorithm for this purpose rather than a SIMPLE algorithm.

The PISO algorithm (Pressure Implicit with Splitting of Operator) is a numerical method developed for pressure-velocity coupling calculation originally intended for non-iterative computation (but later adapted for iterative processes). The algorithm involves a predictor step and two corrector steps and so it can be seen as a SIMPLE algorithm extension by adding a further corrector step (see Figure 11).

The algorithm initiates with the predictor step, in which the discredited momentum equations are given a guessed pressure field in order to yield a guessed velocity field. This procedure is the same as the one used in the SIMPLE algorithm.

Once obtained the resulting guessed velocity field which won't satisfy the continuity equation unless the pressure field is correct, the first field corrector step is started. The guessed pressure and velocity field is used in the continuity equation to obtain the correction field. A second corrector step is needed to solve for a second pressure equation and other discretized transport equations.

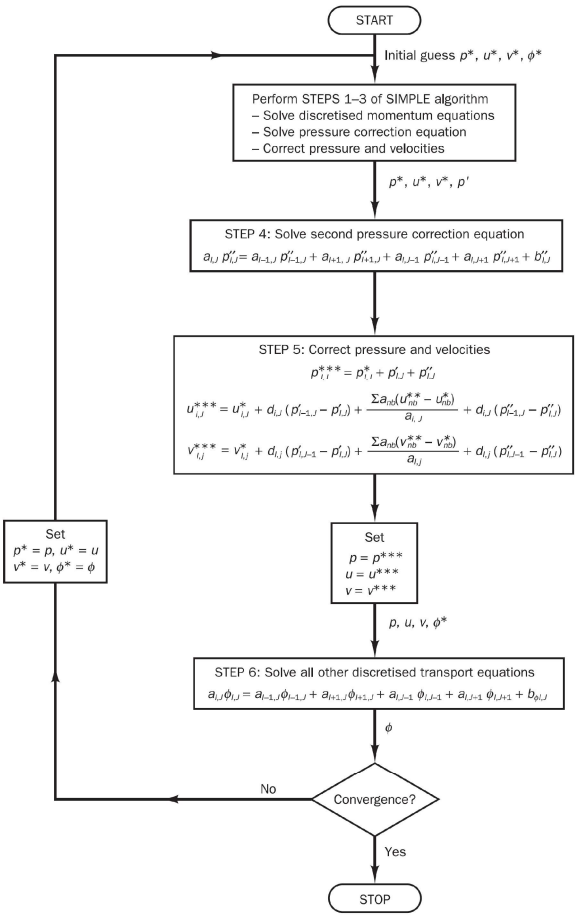

Figure 11: PISO algorithm scheme [19]

3.3.6 MhdFoam solver code description

The mhdFoam solver contains and solves the MHD model equations taking into consideration the magnetic and velocity coupling [20]. As stated before a critical point on the solver operation is to ensure a non-diverging magnetic field for every time step. As stated the projection method was selected by OpenFOAM developers to correct the possible outcome magnetic field into a divergence-free field by using a BPISO correction algorithm. The mhdFoam can be obtained from OpenFOAM installation source on the solver's folder of the installation under the electromagnetic category.

The folder structure of the solver is depicted on Figure 12. The file functions will be briefly explained:

- createControls.H is a header file defining the initialization procedure for the PISO and BPISO algorithms. This header file sets up the algorithm's classes and initializes their parameters such as the relaxation factors for the equations.

- createFields.H is a header file containing the definition of the parameters employed by the solver to load and store the field information about the case to solve. This file will prove important during the solver modification segment of this work in order to let the solver read an external resistivity field.
- The header files readBPISOControls.H, createPhiB.H and magneticFieldErr.H are header files associated with the BPISO algorithm control and should not be modified as the BPISO code on the original mhdFoam solver is already correct.
- The mhdFoam.C file contains the source code written in C++ for the mhdFoam solver functionality.
- The additional folders and files are meant for compilation purposes and should not be modified.

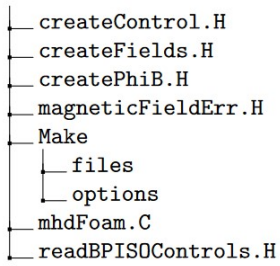

Figure 12: mhdFoam solver organization scheme [20].

Chapter 4: Solver improvement

The improvement of the mhdFoam solver has been performed on the modified mhdFoam code provided by previous works on the topic, and it has been further improved to allow solving for fluids with non-uniform resistivity and viscosity.

In the current chapter, the modifications performed to the code will be explained and justified. The possible shortcomings of the modifications will also be discussed in order to give the reader a further comprehension of what could be achieved by using the modified version of the solver and what problems could arise due to the additional terms included.

The code modification is performed onto the original mhdFoam solver code published on the OpenFOAM API guide [21]. By following the procedure explained here and its reasoning the reader will be able to solve for any geometrical domain with the additional capabilities of having a defined resistivity and viscosity profile.

4.1 Code improvement

In this section the reader will be instructed on how the changes on the original solver were performed in order for the reader to reproduce and inspect them.

4.1.1 Field description

The first step to accomplish the objective was to give the solver the ability to read a custom resistivity and viscosity field from a separated file which will describe the resistivity and viscosity value for each cell centre.

The following code segment found on the createFields.H file located in the solver folder contains the constants which should be read from the transportProperties file of the case intended to solve with mhdFoam. As can be seen the viscosity (named nu in the solver) is defined as a constant for all the cells on the mesh grid and the resistivity is set by a constant parameter DB to all cells equally.

```
1.    Info<< "Reading transportProperties\n" << endl;
2.
3.    IOdictionary transportProperties
4.    (
5.        IOobject
6.        (
7.            "transportProperties",
8.            runTime.constant(),
9.            mesh,
10.           IOobject::MUST_READ_IF_MODIFIED,
11.           IOobject::NO_WRITE
12.       )
13.   );
14.
15.   dimensionedScalar rho
16.   (
17.       "rho",
18.       dimDensity,
19.       transportProperties
20.   );
```

```
21.
22.   dimensionedScalar nu
23.   (
24.       "nu",
25.       dimViscosity,
26.       transportProperties
27.   );
28.
29.   dimensionedScalar mu
30.   (
31.       "mu",
32.       dimensionSet(1, 1, -2, 0, 0, -2, 0),
33.       transportProperties
34.   );
35.
36.   dimensionedScalar sigma
37.   (
38.       "sigma",
39.       dimensionSet(-1, -3, 3, 0, 0, 2, 0),
40.       transportProperties
41.   );
```

The first change to perform is to redefine the constant viscosity constant as a field with a value for every cell on the mesh; for that purpose firstly the current definition of the nu property must be deleted.

```
1.    dimensionedScalar rho
2.    (
3.        "rho",
4.        dimDensity,
5.        transportProperties
6.    );
7.
8.    dimensionedScalar nu
9.    (
10.       "nu",
11.       dimViscosity,
12.       transportProperties
13.   );
14.
15.   dimensionedScalar mu
16.   (
17.       "mu",
18.       dimensionSet(1, 1, -2, 0, 0, -2, 0),
19.       transportProperties
20.   );
```

The new viscosity and resistivity fields are defined as scalar fields (type volScalarField) with the names “nu” and “eta” for viscosity and resistivity respectively and are set to read initially from the files with the same name. An informative text is also set to display when each field is read by OpenFOAM for debugging purposes and the filed I/O options are set so the field must be read at the initial time for the solver to get the resistivity and viscosity values (by using the directive IOobject::MUST_READ) and also, so the solver does not write or overwrite the initial fields unless explicitly stated in the code in order to avoid possible errors (by using the directive IOobject::NO_WRITE).

The following lines defining the new files should be added directly under the constants’ declaration:

```
1.    dimensionedScalar mu
2.    (
3.        "mu",
4.        dimensionSet(1, 1, -2, 0, 0, -2, 0),
5.        transportProperties
6.    );
7.
8.    dimensionedScalar sigma
9.    (
10.       "sigma",
11.       dimensionSet(-1, -3, 3, 0, 0, 2, 0),
12.       transportProperties
13.   );
14.
15.   Info<< "Reading field nu (viscosity)\n"<<endl;
16.   volScalarField nu
17.   (
18.       IOobject
19.       (
20.           "nu",
21.           runTime.timeName(),
22.           mesh,
23.           IOobject::MUST_READ,
```

```
            IOobject::NO_WRITE
        ),
        mesh
    );

    Info<< "Reading field eta (resistivity)\n"<<endl;
    volScalarField eta
    (
        IOobject
        (
            "eta",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::NO_WRITE
        ),
        mesh
    );
```

Now that the solver is already coded to accept the new spatial-dependent variables the new fields must be supplied into the initial time step folder of the case or otherwise the solver won't be able to source the initial resistivity and viscosity conditions. It's also recommended deleting the "nu" definition from the transportProperties dictionary located in constant/transportProperties as it could cause trouble although is no longer needed by the solver in order to initiate the solving procedure.

Finally, as the resistivity and viscosity will be defined by unique variables, the definition for rho, sigma, mu and DB and DBU can be deleted as these variables are only used to calculate constant resistivity and viscosity depending on sigma and rho and this project aims at giving the user the capability of generating a spatial dependent field for resistivity and viscosity.

The DB variable is responsible for magnetic diffusivity (or resistivity as named in this work) and therefore will be replaced by the eta field, so there's no need to keep the variable in the code. The definition of DB in the original code is:

$$DB = \frac{1}{\mu\sigma}. \tag{72}$$

Where μ is the magnetic permeability and σ represents the electrical conductivity of the fluid. These two variables are only used in this parameter and DBU, so there's no need to keep them in the modified solver as they won't be used for any calculation.

The DBU variable is used as a scaling factor depending on the magnetic permeability and density of the fluid for the velocity equation, but in this work the density of the fluid will be considered as the unit and the magnetic permeability as well to ease the data analysis (as all the results will be scaled for density and permeability = 1, they're considered intrinsic properties). The DBU parameter is defined in the original code as:

$$DBU = \frac{1}{\mu\rho \cdot 2}. \tag{73}$$

As stated the DBU parameter will be removed, but it would have a value of 0.5 to account for intrinsic properties. This value will be used in the next section to modify the solver properly.

In order to give the reader a sense of how the eta and nu field files should be configured for the solver to use them constant field syntax is demonstrated for testing purposes.

The field files are named eta and nu with no additional extension (as required by OpenFOAM) and should begin with a standard OpenFOAM header which states configuration parameters such as the text encoding format, the field type and its name. The parameters labelled class and object should be changed accordingly to the magnitude of the field; in the case of resistivity, the object must be set to the variable name “eta” and the class must be set to a scalar field denoted by the volScalarField type.

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      eta;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

Once the header is placed the physical definition of the field is to be set, by using the “dimensions” keyword the variable’s units must be set to the units used in the SI. For resistivity the units are $m^2/_s$ and therefore the dimensions should be set to:

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

The dimensions array indicates the exponent of each fundamental unit of the SI on the target unit. OpenFOAM defines the array as [kg, m, s, K, mol, A, cd], so for a resistivity unit (m^2 s^{-1}) the unit is denoted as [0, 2, -1, 0, 0, 0, 0]. To generate a constant field the uniform directive can be used to state the value of the field in each cell as follows:

```
internalField   uniform (5e-2);
```

The resistivity and viscosity can be configured by changing the internalField parameter for each case. In the following chapter the method to generate non-uniform fields is demonstrated.

The last addition to the field file is the boundary conditions set for the given field. This parameter defines how the magnitude will behave at the boundaries of the geometrical domain and the behaviour can be set individually for each domain surface (patches as defined in OpenFOAM). For this field example the geometrical domain and resulting mesh

are chosen as a cylinder with structured mesh and therefore the boundary conditions should be set for the bases and the side surfaces of the cylinder (named inlet, outlet and sides respectively). The example boundary conditions can be set to:

```
boundaryField
{
    "inlet|outlet"
    {
        type            cyclic;
    }
    sides
    {
        type            slip;
    }
}

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

It is mandatory to state that resistivity and viscosity fields are spatial-dependent but not time-dependent by solver design and therefore the different boundary conditions have little effect on those fields as the solver won't recalculate them. Time-dependent visco-resistive fields are not implemented in the solver design as they're out of the scope of the current project.

4.1.2 Solver code

To modify our solver code to simulate flows for non-uniform visco-resistive fluids the equations regarding these magnitudes will be examined and enhanced accordingly to the physics behind the processes.

The viscosity effect is taken into account in the "fluid part" of the solver, the part in which the Navier-Stokes and continuity equation are solved. The equation is defined in OpenFOAM notation first and then the PISO loop is started to calculate an accurate solution to the pressure-velocity coupling through iteration in a loop. The portion of code of interest is the definition of the velocity equation (UEqn):

```
fvVectorMatrix UEqn
            (
                fvm::ddt(U)
              + fvm::div(phi, U)
              - fvc::div(phiB, 2.0*DBU*B)
              - fvm::laplacian(nu, U)
              + fvc::grad(DBU*magSqr(B))
            );

            if (piso.momentumPredictor())
            {
                solve(UEqn == -fvc::grad(p));
            }
```

The equation UEqn solves the velocity profile (U) by using the model equation:

$$\frac{dU}{dt} + \phi\nabla\cdot(u) - \phi_B\nabla\cdot(2\cdot DBU\cdot B) - \nu\nabla^2\mathrm{u} + \nabla(\mathrm{DBU}\cdot|\mathrm{B}^2|) = 0, \tag{74}$$

The viscosity magnitude in the model equation is a multiplication factor to the laplacian of the velocity field, and as the model equation is computed for each and every cell in the mesh grid the model equation for velocity does not need additional modification in order to be computed

for a non-uniform viscosity field. Nonetheless, as was described previously the DBU variable has been deleted to account for intrinsic properties, so the DBU is replaced by 0.5 which leads to the final model equation. The model equation should be rewritten as the following expression to compensate for the non-present variable:

$$\frac{dU}{dt} + \phi\nabla\cdot(u) - \phi\nabla\cdot(B) - \nu\nabla^2\mathrm{u} + \nabla(0.5\cdot|\mathrm{B}^2|) = 0. \tag{75}$$

And the code for the UEqn expression is then modified to:

```
fvVectorMatrix UEqn
            (
                fvm::ddt(U)
              + fvm::div(phi, U)
              - fvc::div(phiB, B)
              - fvm::laplacian(nu, U)
              + fvc::grad(0.5*magSqr(B))
            );
            if (piso.momentumPredictor())
            {
                solve(UEqn == -fvc::grad(p));
            }
```

The non-uniform resistivity capability is not readily available in the original mhdFoam solver as this solver was designed under the assumption of ideally conducting fluids. To modify the solver to calculate for fluids involving a resistivity field (named eta here) the following steps have been followed.

Originally the mhdFoam solver states the magnetic equation to solve as:

```
fvVectorMatrix BEqn
(
    fvm::ddt(B)
  + fvm::div(phi, B)
  - fvc::div(phiB, U)
  - fvm::laplacian(DB, B)
);
```

Where the DB or magnetic diffusivity coefficient is no longer present in our field definition files. The model equation is stated as follows:

$$\frac{dB}{dt} + \phi\nabla\cdot(B) - \phi_{\mathrm{B}}\nabla\cdot(u) - \mathrm{DB}\,\nabla^2\mathrm{B} = 0, \tag{76}$$

DB is considered in this model as a constant magnetic resistivity which affects the plasma as a whole. In order to enhance the model and change the constant-resistivity effect, the model is replaced for the following one:

$$\frac{dB}{dt} + \phi\nabla\cdot(B) - \phi_{\mathrm{B}}\nabla\cdot(u) - \eta\nabla^2\mathrm{B} - \Big(\nabla(\eta)\times\big(\nabla\times(\mathrm{B})\big)\Big) = 0. \tag{77}$$

The term $\left(\nabla(\eta) \times \left(\nabla \times (\mathrm{B})\right)\right)$ appears due to not considering the resistivity constant and therefore it cannot be neglected on equation as shown during the second chapter of the present work.

The solver is then modified by including the model equation obtained and changing the original equation code to the following:

```
1.   fvVectorMatrix BEqn
2.               (
3.                     fvm::ddt(B)
4.                   + fvc::div(phi,B)
5.                   - fvc::div(phiB,U)
6.                   - fvm::laplacian(eta,B)
7.                   - ((fvc::grad(eta))^fvc::curl(B))
8.               );
```

Given the made changes to the solver, which will be referred in this document as NonUniformMhdFoam, it becomes necessary to test the solver in search for issues induced by the addition of the non-uniform visco-resistive capabilities. A fundamental value which should be taken into account when validating the code is the magnetic flux divergence as calculated at the end of each timestep.

The BPISO algorithm iterates over the solution values for the magnetic field and corrects it to adjust the magnetic flux divergence of the solution to 0. As it becomes numerically impossible to grant a magnetic flux divergence of zero, a very low threshold is set in order to ensure a low magnetic divergence. A possible inaccuracy in the code due to the introduced modifications could lead to a non-solvable system because a higher magnetic divergence than the given threshold. This effect will be of great importance when analyzing if the solver is usable or not.

Additionally, the solver is modified to solve for 2 different magnetic fields, the actual magnetic field experienced inside the geometrical domain is labelled *B*, and it is composed by the sum of a fixed magnetic field B_{Fix} and a variable magnetic field B_{ble}. The fixed magnetic field corresponds to the externally imposed magnetic field on the domain, such as the toroidal field imposed by the coils of a Tokamak. The variable magnetic field is the magnetic field induced in the plasma by the outer imposed magnetic field. Finally, a new field is defined as the sum of both imposed and induced fields as the resulting B field:

$$B = B_{Fix} + B_{ble} \tag{78}$$

This modification is performed on the solver equation by changing the model equation to:

$$\frac{dB_{ble}}{dt} + \phi\nabla \cdot (B_{ble}) + \phi\nabla \cdot (B_{Fix}) - \phi_{\mathrm{B}}\nabla \cdot (u) - \eta\nabla^2 B_{ble} - \eta\nabla^2 B_{Fix} - \left(\nabla(\eta) \times \left(\nabla \times (B_{ble})\right)\right) - \left(\nabla(\eta) \times \left(\nabla \times (B_{Fix})\right)\right) = 0. \tag{79}$$

The model is implemented on the solver by changing the model equation and applying the total B field calculation. The final code is modified by changing the previous model equation to:

```
fvVectorMatrix BEqn
(
      fvm::ddt(BVble)
    + fvm::div(phi,BVble)
    + fvc::div(phi,BFixed)
    - fvc::div(phiB,U)
    - fvm::laplacian(eta,BVble)
    - fvc::laplacian(eta,BFixed)
    - ((fvc::grad(eta))^fvc::curl(BVble))
    - ((fvc::grad(eta))^fvc::curl(BFixed))
);
BEqn.solve();
B=BVble+BFixed;
```

Additionally, the BFixed and BVble fields have been added to the createFields.H header file to allow the solver to load the fields and perform the calculations.

4.2 Solver schemes

The solver schemes are defined as the numerical methods used by OpenFOAM when performing the calculus detailed in the solver code. Solver schemes are available for a wide range of operations and are provided by OpenFOAM libraries, and so they'll be treated as extensively tested code (only the schemes provided by the official OpenFOAM distribution are considered).

The first tests of the non-uniform solver resulted in solver crashes due to increasingly higher magnetic flux divergences. Different solver schemes were proposed by the tutor as a possible alternative to the chosen ones in order to improve numerical accuracy and prevent the magnetic divergence.

The possible solution schemes options include:

- Time schemes: Provide the numerical schemes used to solve first and second order time derivatives. The schemes available for the time derivatives include setting the derivatives to zero (steadyState scheme), transient schemes of first order (Euler and backward schemes) and second order transient derivatives (CrankNicolson scheme).

  The time derivative schemes were set to Euler at first, giving a first order numerical solution, but were later changed to CrankNicolson with an off-centreing coefficient of 0.9 (as indicated in the OpenFOAM guide) due to inaccuracy found in the solution.

As the modified (and the original) solver only considers differential equations with first order time derivatives the setting for second time derivatives is not relevant for this work.

- Gradient schemes: Specifies the numerical solving schemes used to solve gradient operator. The default scheme is Gauss linear, which specifies the standard finite volume discretization of Gaussian integration which interpolates the values from cell centers to the face centers. The other possible scheme is Gauss cubic scheme, which is a third-order scheme which higher numerical accuracy, which is the one used in the configuration of the modified mhdFoam solver.

- Divergence schemes: The interpolation scheme is selected from the full range of schemes, both general and convection-specific. The choice critically determines numerical behaviour. The syntax here for specifying convection-specific interpolation schemes does not include the flux as it is already known for the particular term. The divergence scheme employed in the proposed cases is the cubicCorrected schemes which provides a fourth order unbounded divergence numerical interpolation.

- Laplacian schemes: The schemes used for computing the laplacian operator in the solver code. The laplacian operator, defined as:

```
1.  laplacian(A,B)
```

Corresponds to a mathematical formulation of the laplacian operator give as:

$$\nabla \cdot (A \nabla B). \tag{80}$$

The operator only accepts Gauss as a numerical scheme, but requires the selection of additional numerical schemes for the interpolation of the diffusion coefficient A and a surface normal gradient coefficient for the gradient operator ∇B.

The syntax for the Laplacian numerical scheme is selected as Gauss followed by the interpolationScheme and the snGradScheme.

```
1.  Gauss <interpolationScheme> <snGradScheme>
```

In the setup for the modified solver, Gaussian laplacian scheme is selected with linear and corrected schemes as interpolation and snGrad schemes.

- Interpolation schemes: The interpolation schemes defines the numerical scheme used to interpolate the cell centre values for a field to the face-centre values. In the present solver case the interpolation schemes are important for the velocity and magnetic fields, which need to be interpolated in order to calculate the magnetic and velocity flux across the cell faces.

- Surface Normal Gradient schemes: A surface normal gradient is evaluated at a cell face; it is the component, normal to the face, of the gradient of values at the centers of the 2 cells that the face connects. The option chosen for the modified mhdFoam solver is to use fourth order as it'll provide the best solution accuracy possible among the other available schemes as well as compensate for cell non-orthogonality.

It is worth mentioning that changing the numerical solver schemes has the potential to influence the solution accuracy and therefore become a key parameter to validate if the results obtained in this work are valid or not. The first try at solving the cases was made using the lowest-order solution schemes and resulted in the solver heavily divergence and not being able to yield any solution. Upon this finding the author was instructed to change the solution schemes used in the solving process to higher order ones by switching to the mentioned options.

Chapter 5: Preprocessing

In this section the two different geometrical domains used for solver testing will be exposed and explained for the reader to see how the different cases were set up and simulated with the exiting OpenFOAM mhdFoam solver and the enhanced solver which is the object of the present work.

5.1 Meshing procedure

In order to create a run a computational fluid dynamics simulation firstly we do need to define a geometrical domain in which the simulation will take place (where the fluid will flow).

In computational fluid dynamics a geometrical domain is defined by means of a mesh, which is a set of interconnected cells that share faces and nodes among them and create a discrete representation of the geometry of the selected domain. Model equations for the fluid and phenomena are then applied as if the fluid was contained in each cell, calculating the flow across the cell boundaries.

A good simulation is influenced by the mesh of its domain; the rate of convergence of the solution, the results accuracy or the computing time required for the calculus is directly influenced by the quality of the mesh defined by several parameters which will be briefly covered on the next section.

The meshes in OpenFOAM are represented as a collection of nodes and faces located in separate files under the folder constant/polymesh. The node's and cell centre's coordinates are related to a Cartesian coordinate system which origin and directions are set during the meshing procedure.

5.1.1 Meshing software

Once a domain has been selected, a software application is needed to procedurally generate a mesh with the given boundaries of the given domain. Several applications exist for that purpose and to select the right option the mesh type to be generated has to be known beforehand.

The meshes can be of two main types:

- Structured meshes: These meshes are constructed using regular cells which are stacked in an ordered fashion, making them easily describable and computed by simple software tools. These meshes are preferred over unstructured ones as they provide better means of structure control.

Nonetheless, structured meshes are not well fitted for complex geometries as their structure does not provide good adjustment to surfaces more complex than planes and simple curves. This kind of meshes are only fitted to be used in very simple geometrical cases and as a mesh basis for further software tools to shape them to a more complex geometry.

- Unstructured meshes: These kinds of meshes are usually automatically generated by meshing software and are much more complex than their counterparts. The cells in these meshes are not placed regularly but by using computing algorithms which select the best fit for a cell structure and volume given the position in the geometrical domain.

  When complex geometrical domains are required these meshes are the only option available, as no regular mesh can shape to the desired geometry. These meshes can be obtained by using shaping software over a regular mesh and a desired geometrical model. Then the shaping software tailors the initial structured mesh to an unstructured regular mesh which fits the given geometry; further refining steps can be applied to the resulting mesh to increase fitness and resolution in given areas.

The software packages used during the development of this project will be listed and briefly explained.

BlockMesh

BlockMesh is a software package provided with OpenFOAM distributions which provides regular mesh generation capabilities by using a simple syntax dictionary. As it is a built-in tool in the OpenFOAM package it's designed to generate meshes in the polyMesh format adequate for the simulation case run with no further steps.

BlockMesh has been used during the present work to produce the base meshes for the toroidal domain and the desired meshes for the cylindrical tests.

The tool is based on the generation of geometries by using blocks. A block is defined as a spatial region designated as an orthohedron by 8 vertices and which is tessellated by hexahedral cells. The faces of the block can be shaped into planar surfaces (default setting) or curved surfaces by definition of arcs (and thus being parametric curved surfaces).

In order to generate moderate complex geometries several blocks can be joined and shaped to compose a multi-block geometry. The different blocks must have at least one matching face which will be interpreted by blockMesh as a joining face between blocks. Matching aces for blocks are automatically fetched by blockMesh and the cell joining is performed; nonetheless the matching faces on the adjacent blocks must have the same cell structure

and as otherwise the mesh would become unstructured during the joining process. Such geometry can be used to generate a cylindrical domain which will be exposed next.

SnappyHexMesh

SnappyHexMesh is a mesh tailoring software provided with the OpenFOAM package which does not have explicit mesh generation capabilities; its purpose is tailoring a base mesh (possibly generated by blockMesh) into a given geometrical domain defined in a surface definition file (STL).

The software package tailors the mesh (which must be larger than the desired domain) by removing the cells outside the desired domain. The package also offers options to shape the resulting mesh into the desired domain by shaping the faces of the cells to the geometry surface and also provides mesh refinement capabilities, which prove useful when more resolution is needed in certain regions, for example on cells adjacent to the surface boundaries on the geometry, which are sensible of experiencing boundary conditions resolution issues if not studied carefully.

The meshes produced by SnappyHexMesh consist of hexahedral cells arranged in a non-structured fashion and are readily available for OpenFOAM computing as they're generated in polymesh format.

SnappyHexMesh is considered as a complex mesh tailoring software with great performance in complex mesh generation. It is extensively tested and documented and therefore it is selected as a tool to generate complex domains such as the toroidal domain exposed in this project.

Gmsh

Gmsh is an open source finite element mesh generator which has parametric input mesh generation capabilities and a wide range of parameters to generate the desired meshes. It has been selected in this work due to its unstructured meshing capabilities for the cylindrical domain mesh generation.

Gmsh provides the user a graphical environment to parametrically design and generate complex meshes. The geometrical domain can be input as a set of faces defined from a group of simple 2D geometrical shapes such as points, lines, arcs and curves. The geometrical information is stored in a geo file which serves as a definition of the domain which is later meshed by the tool by adjusting cell parameters such as predominant cell shaping, target cell size and possible regular tessellation.

The produced mesh file (eMsh) has to be converted to polymesh format prior to being used

by OpenFOAM; this task can be accomplished by using the built-in mesh translator *GmshToFoam.*

5.1.2 Mesh quality parameters

Meshing a geometrical domain is no easy task and several problems can arise from incorrectly generated meshes which will translate into bad solutions or even unsolvable cases [22].

Mesh coarseness

When meshes become too coarse for the case parameters and the simulation the solver may tend to a non-convergent solution which can cause the solver to crash or arrive at a non-physical solution. On the other hand finer meshes tend to be much more resource-hungry as more cells need to be individually solved and therefore more calculations need to be done to arrive at a solution.

The usual procedure regarding mesh coarseness is to identify the areas in which more resolution is needed, being those the areas of interest and those that exhibit a special characteristic such as high velocities, being close to boundary layers and geometrical singularities. Upon specific regions being selected for finer mesh generation a process of mesh refinement can be performed to improve mesh resolution. SnappyHexMesh provides means of performing these operations.

Aspect ratio

The aspect ratio is a measure of how a cell is deformed from its “ideal” shape. For example an ideal hexahedron is that with equal faces and effectively a cube; whereas in non-structured hexahedron meshes the hexahedron cells are deformed, for example a hexahedron which base’s side is much smaller than its height.

Each cell shape (tetrahedron, hexahedron, prism ...) has its own aspect ratio calculation method which is correctly applied by mesh diagnostic software and provides a measure of how the cell’s shape are deformed from an ideal cell. The aspect ratio of the mesh is expressed as the mean aspect ratio among the cells which compose the mesh and the maximum and minimum aspect ratio in the mesh. It should be kept as close as 1 as possible but for non-simple geometries the cells will surely experiment deformation, so an aspect ratio equal to the unity is non-achievable.

Orthogonality

This measure corresponds to the angle calculated between the vector connecting two

adjacent cell's centroids and the surface connecting the two control volumes through which the vector crosses the control volume. It's a measure of how the cells are deformed from an ideal cell shape in which the interest surface is always perpendicular to the vector. The closer the measured value is to 90 degrees the lesser the errors will be induced into the solution by an incorrect interpolation on the cell's flux crossing the given surface.

The range of this measure comprises 0 to 180 degree and should be kept as close to 90 as possible.

Skewness

Skewness of the cells is defined as the difference between the shape of the cell and the shape of an equilateral cell of equivalent volume. [22]When skewness is high the accuracy of the solution is decreased and may lead to a solver failure or inaccurate results.

Smoothness

Smoothness is a measure of how cell volume changes among adjacent cells. It is normal for a refined mesh to change cell volumes as it approaches a specific mesh region as meshing algorithms do need to place larger or smaller cells in certain locations in order to mesh the specified geometry, but placing large cells adjacent to very small cells can cause errors during simulations as flows across the cell's surfaces can be altered by numerical inaccuracy in adjacent cells with highly varying volume.

This variable describes the volume ratio between two adjacent cells which share a common face. The ratio is calculated as the ratio of the small volume to the larger one and therefore having a possible range between 0 and 1. It is always desired to have a mesh smoothness closer to 1 and avoid having cells with despairing volumes; when cells with much smaller volume are needed a gradual volume diminution is preferred to sudden volume changes in adjacent cells.

These parameters are taken into account when creating a new mesh for case validation as the results from the simulation cannot be considered valid when the mesh itself poses a inaccuracy threat to the solution. Often if these mesh quality parameter are non-controlled the numerical solution may seem valid when it is clearly inaccurate due to bad meshing.

Following, the meshing, geometrical and field setup of the different test cases is discussed.

5.2 Cylindrical domain

The cylindrical domain was the first domain created and simulated with OpenFOAM due to its geometrical simplicity compared to other domains such as the toroidal or stellator domain (which was discarded as a domain of interest due to its complexity).

The proposed domain was set as a cylinder with a radius of 1m and a length of 8m. This geometrical domain is then meshed through meshing application software in order to obtain a gridded domain in which the simulation will take place.

The individual mesh generation process for each individual mesh case is discussed as different meshes will have different performance and effects on the solution. The field setup is discussed only once for every field configuration as it only depends on the geometrical domain and is calculated from the cell centre coordinates (extracted and computed for each individual mesh).

This case posed serious meshing issues as demonstrated by previous work [2] in which the cylindrical domain was attempted to mesh but became quickly discarded due to visible numerical inaccuracies and noise derived from the mesh itself as will be demonstrated.

It is worth mentioning that the patches (surfaces as defined by OpenFOAM) considered for this mesh were the “inlet” and “outlet” as the bases of the cylinder and the side of the cylinder as “sides”. The “sides” patch was always set to “patch” type in the boundary file located in constant/polyMesh; but the “inlet” and “outlet” patches were set either to “cyclic” type or “cyclicAMI” type.

For the unstructured meshes (Snapped and Unstructured) the patch type for the bases ought to be set as “cyclicAMI” as the faces of the cells on the inlet and outlet surfaces could not match as a result of unstructured meshing, but the type was set to “cyclic” on the structured meshes (O-grid and axisymetric) as their structured nature ensured matching cell faces.

It is important to mention that cyclicAMI type patches incur in 2D value interpolation for velocity and magnetic field fields on the inlet and outlet faces in order to allow flux calculation among cells with non-matching faces. The possible inaccuracy of this interpolation has not been taken into account for the result analysis.

5.2.1 O-Gridded cylinder

The first mesh attempt consisted in a simple hexahedral dominant mesh built in a O-grid configuration, where the centre of the cylinder is defined by a hexahedral domain and the external faces of the domain are defined by trapezoidal domains shaped to the cylindrical outer surface. The aim of this mesh is to generate a structured cylindrical mesh which eases

its definition procedure without compromising the aspect ratio and the smoothness of the grid.

The central hexahedral block provides a better cell transition to the external blocks than that accomplished by using an axisymetric mesh due to the cell volume being almost constant at the central block (when defined as an hexahedron with a square base) and adapting nicely to the cells on the outer blocks of the mesh.

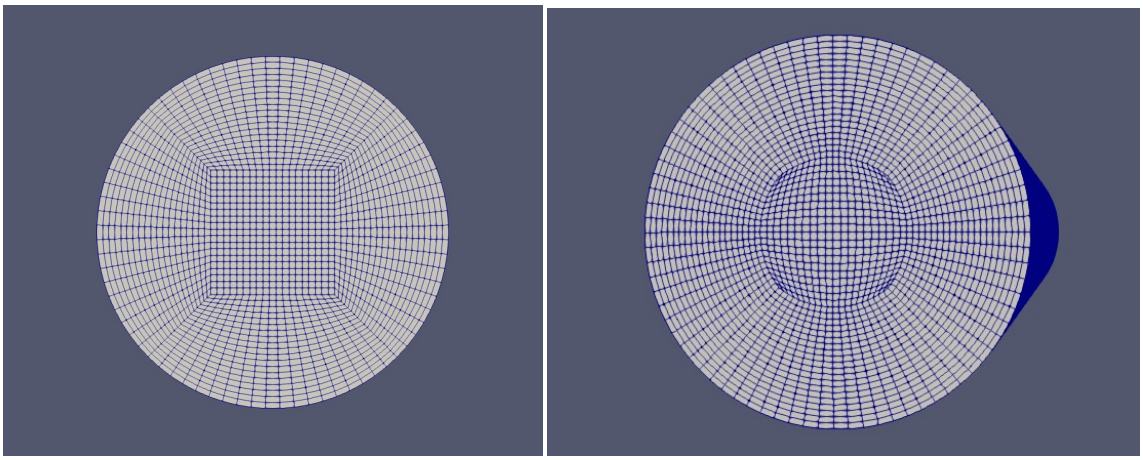

Figure 13: O-grid mesh generated by using blockMesh, front view. Several mesh variants for the O-grid mesh topology have been proposed with same results.

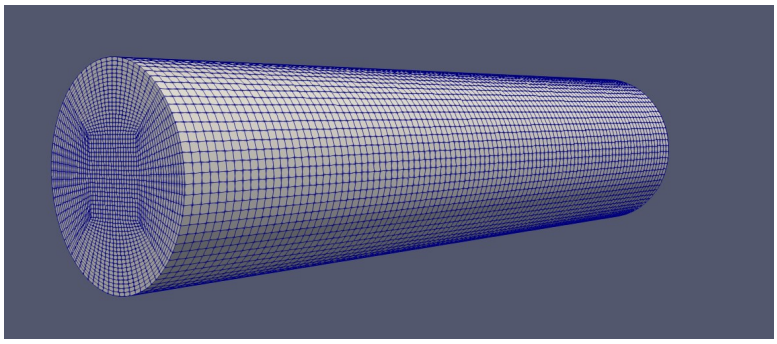

Figure 14: O-grid mesh generate by using blockMesh, side view.

The orthogonality achieved on the join between two outer blocks differs from the ideal 90 degree value but is considered good enough for the calculation to be valid (insert value). The mesh can be further improved by changing the inner block planar surfaces to curved surfaces in order to achieve better orthogonality between the inner and the outer blocks on the boundary between them.

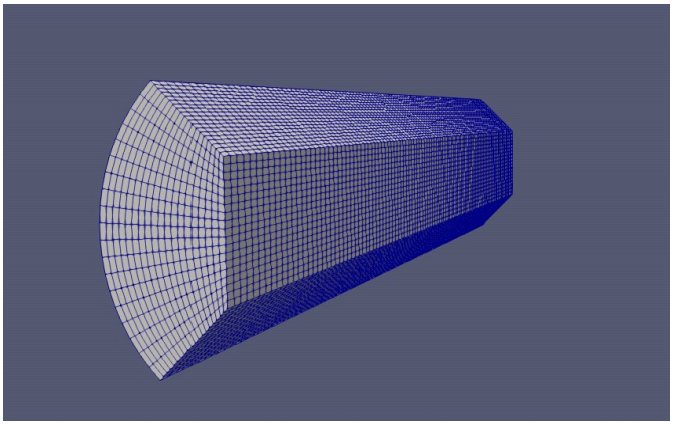

Figure 15: Outer block of the O-grid mesh, generated with blockMesh.

A critical point in the mesh is found on the intersection of two outer blocks and the inner block, where the hexahedral cells become heavily deformed and which will prove to be a region of high numerical error.

The mesh quality can be further increased by dividing the outer blocks into 8 or 16 in spite of the proposed 4 and by creating an intermediate ring of trapezoidal blocks between the central block and the outer blocks. By increasing the number of blocks, the gradient of cell shaped can be smoothened and the mesh can be improved. This method has proven inefficient in increasing the mesh quality due to the numerical errors appearing at the boundaries of the blocks, which suggests a possible mesh-generation related problem and will be discussed in the results chapter of this project.

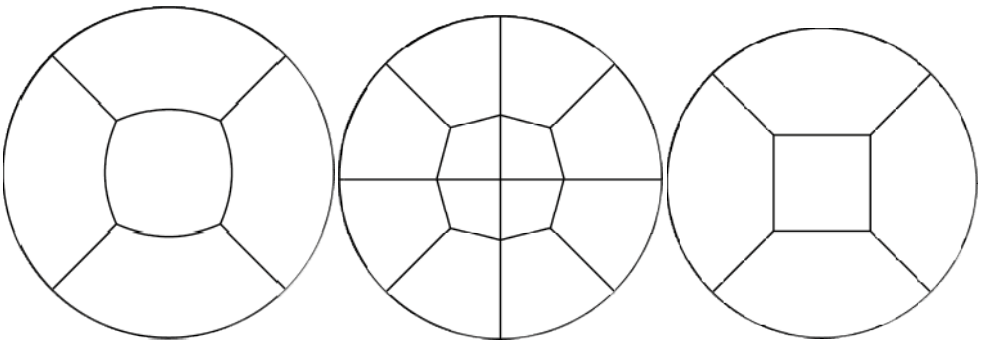

Figure 16: Schematic of three variants of the O-grid mesh topology. All of them were tested.

5.2.2 Axisymetric cylinder

The axisymetric cylinder domain is the easiest approach to the cylinder meshing case and also proved to be the least performant.

The mesh is structured into at least 4 hexahedral blocks with triangular bases (two of each block's vertex are the same). The blocks present as triangles with the outer surface rounded to achieve a cylindrical outer surface.

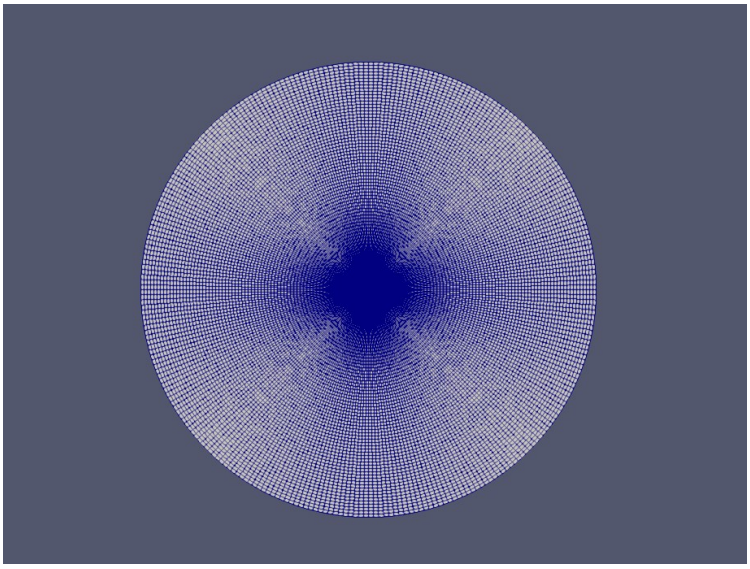

Figure 17: Axi-symmetric cylindrical mesh, front view.

This mesh has the major drawback that, as it is a structured mesh created by blockMesh, the number of nodes in the block's edges for a given face have to be equally for faced edges. That condition forces the cells to tend to a 0 volume at the centre of the cylinder.

While the gradient of cell volume is consistent and no very despairing volume cells are adjacent the aspect ratio of the cells and its skewness becomes dramatically worsened and the mesh is surely unusable, but it will prove useful when analyzing why the structured mesh on the cylinder domain repeatedly fails.

5.2.3 Snapped cylinder

The snapped cylinder mesh is a mesh built by using the snappyHexMesh utility. The mesh is obtained by firstly generating a simple hexahedral mesh of 1.2x1.2x8m which is then fed to the snappyHexMesh utility together with an STL file containing the surfaces of the desired cylinder.

Several iterations over this method have been performed and success has been achieved when using STL files only containing raw surfaces with no volume. Those surfaces have been obtained by using a parametric CAD software, OpenSCAD, and the exported STL file has been edited by a python script in order to delete all the volume inside the cylindrical domain and leave only the surface information, being that the three surfaces that comprise the geometrical domain (two bases named inlet and outlet and a cylindrical surface named

sides).

When using an STL file describing a cylindrical volume snappyHexMesh has proven inefficient on fitting the base mesh generated by BlockMesh to the desired shape, especially on the base edges.

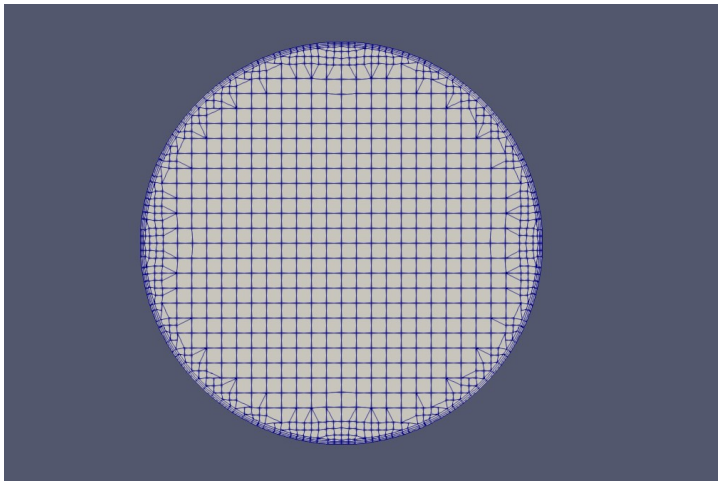

Figure 18: Snapped mesh, generated by blockMesh and snapped by using SnappyHexMesh, front view.

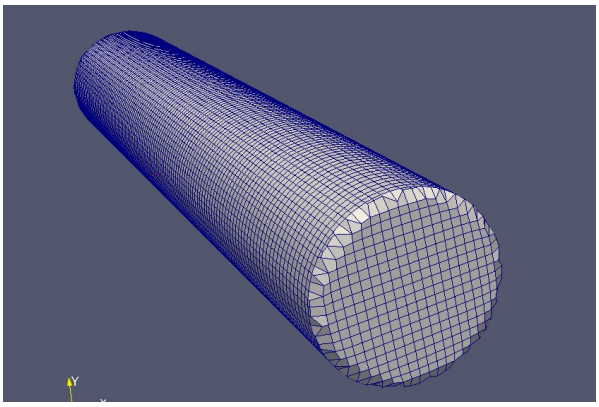

Figure 19: Snapped mesh, side view.

SnappyHexMesh was configured to keep the mesh portion inside the cylinder and discard the cells outside the domain. Then the utility was instructed to fit all the cells laying at the boundaries of the surface to the cylindrical surfaces and perform and optimization procedure to grant good mesh parameters. It was also programmed into the snappyHexMesh dictionary to perform a layer laying process on the surface boundaries of the cylindrical shape in order to achieve better resolution and accuracy on the boundary layer.

5.2.4 Unstructured cylinder

Finally, as the last meshing procedure for the cylindrical domain, a unstructured mesh was

proposed as a possible solution to the meshing problem of the domain. As the issue which will be explained in the results chapter was thought to be caused by inappropriate meshing join between block mesh blocks, a fully algorithm-generated unstructured mesh could prove to provide at least a non-numerically related problem on the mesh by eliminating the block joining procedure in the generation.

The basic block geometry was introduced into Gmsh as a collection of surfaces and points parametrically generated and then a fine mesh was generated by using the unstructured tet-dominant default algorithm provided in Gmsh.

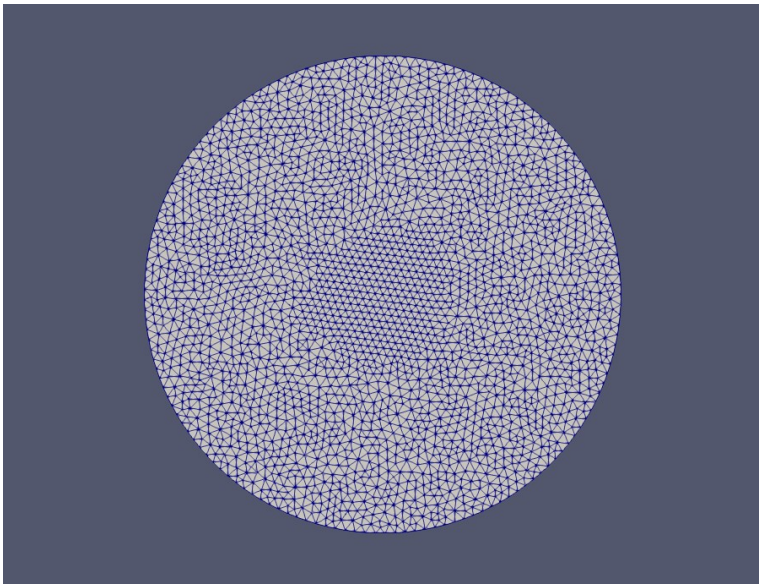

Figure 20: Unstructured mesh generated by using Gmsh, front view.

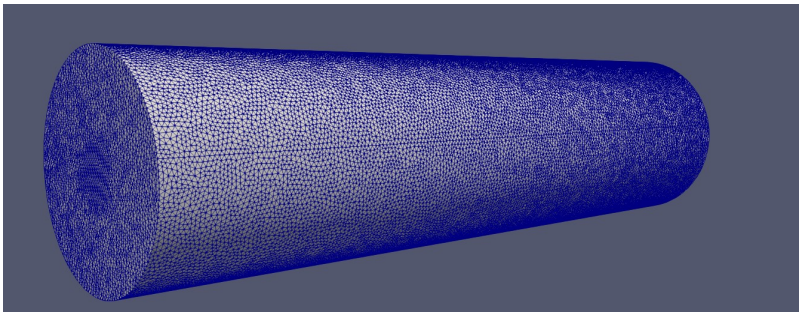

Figure 21: Unstructured mesh generated with Gmsh, side-view.

The resulting mesh proved to be well fitted into the cylindrical domain and solvable by the mhdFoam solver in the conditions explained in the next section. This mesh although neither perfect nor recommendable provided insight in what the issue in the cylindrical domain could be.

The results regarding this mesh will be further discussed in the results chapter.

5.2.5 Fields set up

For the field creation on the cylindrical domain the coordinate system is located at the canter of the inlet face with the z positive axis of the Cartesian coordinate system so the height of the cell in the cylinder is directly obtained as the z coordinate, while the radius of the cell coordinate and the angle respective to the x axis can be computed as:

$$h = z, \tag{81}$$

$$r = \sqrt{x^2 + y^2}, \tag{82}$$

$$\theta = \tan\left({}^{y}/_{x}\right)^{-1}. \tag{83}$$

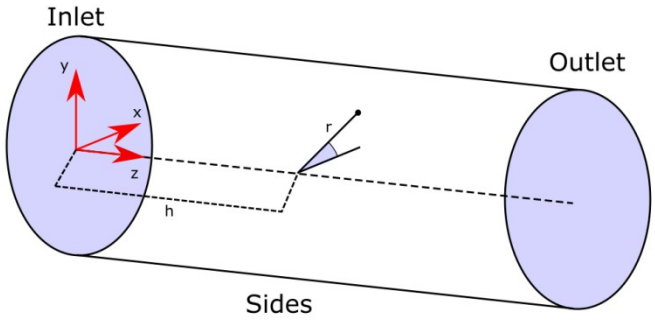

Figure 22: Cylindrical geometrical domain scheme. Source: own work

Given the coordinate system for the cylindrical domain in terms of (h, r, θ), the fields used for the case set up can be obtained independently of the chosen mesh as:

- Initial velocity field: The selected velocity field for plasma simulations is chosen as a randomly generated velocity vector given by a uniform distribution between 0 and 1e-6 m/s. This velocity profile emulates a non-structured plasma ideal as the initial time step from which the simulation will evolve. By ensuring a random velocity distribution one can ensure no structure is induced into the simulation by the field set up and thus the resulting field structure will be the result of the fluid simulation.

  The velocity field is then described as

$$\vec{u} = (u_x, u_y, u_z) = \left(1 \cdot 10^{-6} \cdot U(0,1), 1 \cdot 10^{-6} \cdot U(0,1), \ 1 \cdot 10^{-6} \cdot U(0,1)\right). \tag{84}$$

being $U(a,b)$ a continuous uniform distribution in the closed interval $[a,b]$ and (x,y,z) the coordinates of the cell's centre point.

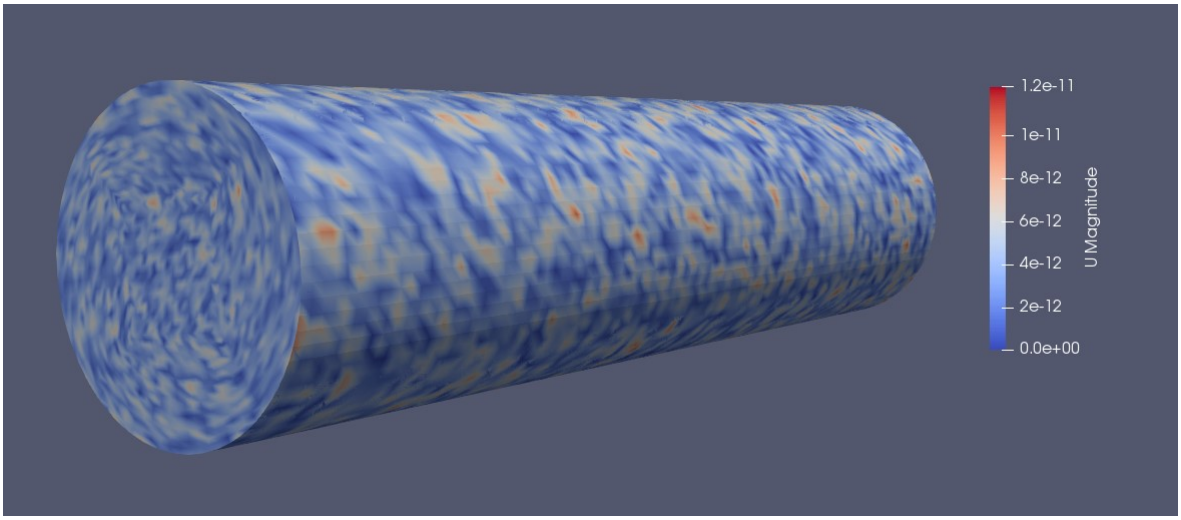

Figure 23: Initial velocity profile for the cylindrical mesh (O-grid mesh).

- Initial magnetic field:

Poloidal magnetic field

The poloidal magnetic field configuration set up configuration as a Poloidal magnetic field is mainly used in this work as a test bench for the cylindrical domain. The magnetic field configuration consists in an axial magnetic field which is constant throughout the cylinder volume and parallel to the z axis of the coordinate system:

$$\overrightarrow{B_z} = B_z\hat{z}. \tag{85}$$

In this configuration there's another magnetic field component, referred as the poloidal field, this field component is perpendicular to the axial component, and therefore to the z axis of the coordinate system. This poloidal field always spins in the same direction throughout the cylinder volume and is zero at the centre and maximum at the cylinder surface. The poloidal component of the magnetic field can be mathematically produced for a cylinder given each cell's centre coordinates (x,y,z). The magnitude of the component is proportional to the distance to the z axis of the cylinder, and therefore can be computed as:

$$\left|\vec{B}_P\right| = \frac{B_\theta}{R}\sqrt{x^2+y^2}. \tag{86}$$

With B_θ being the maximum magnetic field magnitude which is achieved on the surface of the cylinder. The direction of the poloidal field is extracted from the (x,y) vector by rotating it 90 degrees clockwise or anticlockwise depending on the field setup.

$$\widehat{B_P} = \frac{-y}{\sqrt{x^2+y^2}}\hat{x} + \frac{x}{\sqrt{x^2+y^2}}\hat{y}; \tag{87}$$

The final magnetic field generated is described as:

$$\vec{B}_0 = \overrightarrow{B_z} + \vec{B}_P = B_z\hat{z} + \left|\vec{B}_P\right| \widehat{B_P}\,. \tag{88}$$

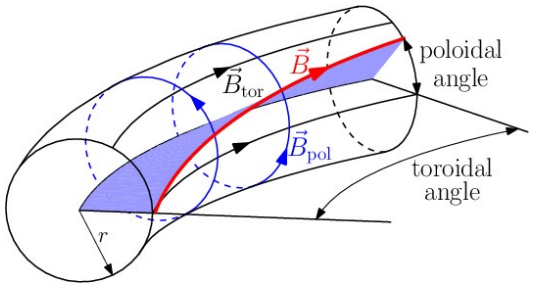

Figure 24: Toroidal geometrical domain scheme [23].

The ratio between the axial and poloidal magnetic field is named pinch ratio and is a parameter of fundamental importance in the present work, the cases will be studied by scanning the parameter as a function of the pinch ratio to see the evolution of the solution.

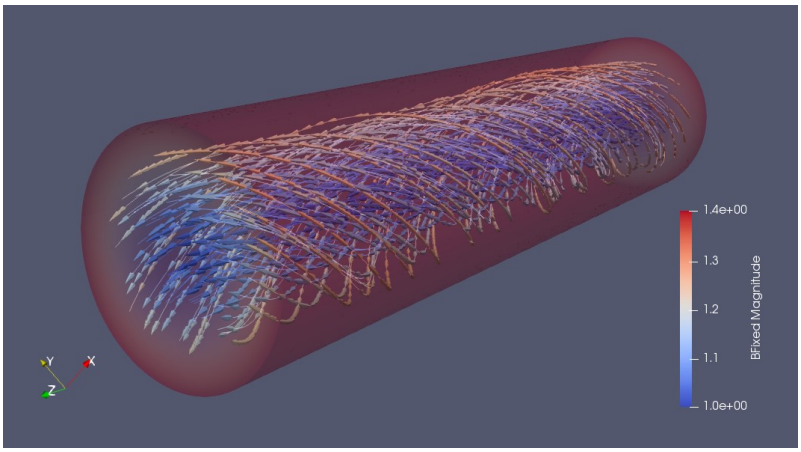

Figure 25: Initial externally imposed magnetic field for the cylindrical domain case.

The external magnetic field configuration is obtained via a custom python script which reads the cell centers coordinates from a file located in the 0 folder of the case and

which contains the cartesian coordinate of the each cell centre. The script performs the given calculus and computes the magnetic field vector components for each cell, which are later written into 0/BFixed magnetic field description file.

Mirror machine-like magnetic field configuration

The externally forced magnetic field configuration for a mirror-machine like cylindrical domain can be obtained by following mathematical expression, in which the axial magnetic field B_z and the poloidal magnetic field B_r are described and obtained as:

$$B = B_z + B_r, \tag{89}$$

$$\nabla \cdot B = 0 = \frac{1}{r}\frac{\partial}{\partial r}(rB_r) + \frac{\partial B_z}{\partial z}, \tag{90}$$

$$\frac{\partial}{\partial r}(rB_r) = -r\frac{\partial B_z}{\partial z}; \tag{91}$$

Considering that $\frac{\partial B_z}{\partial z}$ is given at $r = 0$ and does not vary much with r, then:

$$rB_r = -\int_0^r r\frac{\partial B_z}{\partial z}\,dr \approx -\frac{1}{2}r^2\left[\frac{\partial B_z}{\partial z}\right]_{r=0}, \tag{92}$$

$$B_r = -\frac{1}{2}r\left[\frac{\partial B_z}{\partial z}\right]_{r=0}; \tag{93}$$

The axial magnetic field (B_z) is produced by a single coil in the simplest approach and therefore it can be described as following (with R being the radius of the coil):

$$B_z = \frac{\mu_0 R^2 I}{2(z^2 + R^2)^{2/3}}; \tag{94}$$

Therefore $\frac{\partial B_z}{\partial z}$ can be obtained as:

$$\frac{\partial B_z}{\partial z} = -\frac{2\,\mu_0 R^2 I\, z}{3(z^2 + R^2)^{5/3}}; \tag{95}$$

Then the magnetic field equates as:

$$\vec{B} = \frac{\mu_0 R^2 I}{2(z^2 + R^2)^{2/3}}\hat{z} + \frac{\mu_0 R^2 I\, z\, r}{3(z^2 + R^2)^{5/3}}\hat{r}; \tag{96}$$

But on the common magnetic mirror machines, the B_z field is created by two inline coils with the same current direction separated a distance L_z. The field created by that configuration is (assuming both coils with the same radius and current):

$$B_z = \frac{\mu_0 R^2 I}{2(z^2 + R^2)^{2/3}} + \frac{\mu_0 R^2 I}{2((z - L_z)^2 + R^2)^{2/3}}, \tag{97}$$

$$\frac{\partial B_z}{\partial z} = -\frac{2\,\mu_0 R^2 I\, z}{3(z^2 + R^2)^{5/3}} - \frac{2\,\mu_0 R^2 I\,(z - L_z)}{3((z - L_z)^2 + R^2)^{5/3}}; \tag{98}$$

And therefore the B field is:

$$\vec{B} = \left(\frac{\mu_0 R^2 I}{2(z^2 + R^2)^{2/3}} + \frac{\mu_0 R^2 I}{2((z - L_z)^2 + R^2)^{2/3}}\right)\hat{z} + \left(\frac{\mu_0 R^2 I\, z}{3(z^2 + R^2)^{5/3}} + \frac{\mu_0 R^2 I\,(z - L_z)}{3((z - L_z)^2 + R^2)^{5/3}}\right) r\,\hat{r}. \tag{99}$$

The final magnetic-mirror like machine magnetic field configuration is obtained by solving the latter field equation given the current circulating through the coils, the separation distance among the coils and each cell coordinate on the grid. The result obtained for a current of 10 Amperes, a separation of 8m and a cylindrical domain mesh can be observed in Figure 26.

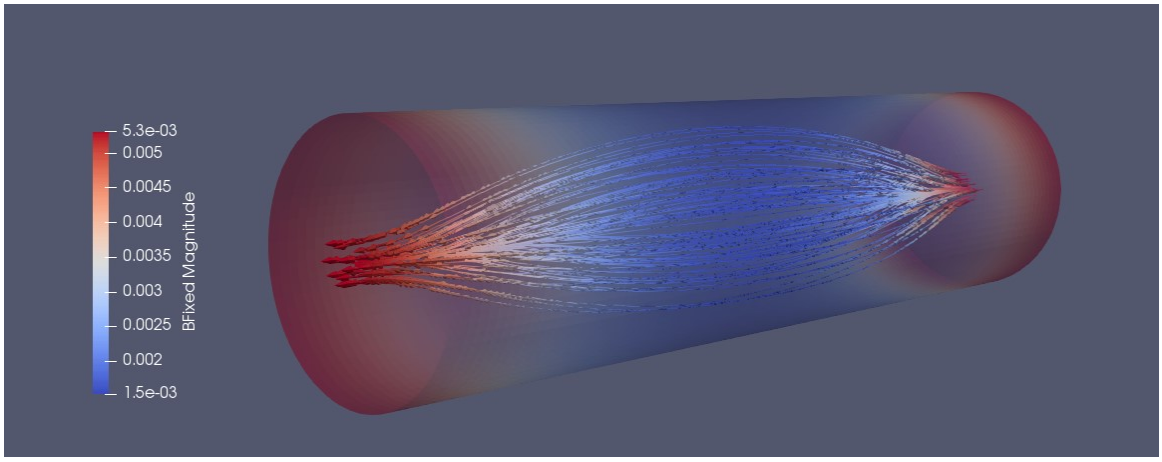

Figure 26: Magnetic mirror externally imposed magnetic field configuration for the cylindrical case.

5.3 Toroidal domain

The toroidal domain presented in this work as a test domain for the modified solver is selected as the best option among the possible domains due to its good performance in previous works carried out on the original solver [2].

The toroidal shape seeks to emulate that of the Tokamak fusion reactors in a simple manner and thus provide some motivation towards the development of this work in nuclear fusion research. Although the toroidal domain is said to resemble the shape of a type of fusion reaction machines, the author is aware that the shape of a Tokamak is much more complex and therefore the results obtained for the toroidal domain aren't in any case applicable to any tokamak related research.

The discussed toroidal domain and meshing is greatly inherited from previous works which proved to work and solve well [2].That meaning the mesh used has only been lightly improved from the previous work.

The geometrical domain consists in a toroidal shape with a single patch which encapsulates the whole domain as a surface (named sides). The patch type is set to "patch" in the boundary dictionary located in the constant/polyMesh dictionary as no cyclic condition should be met. The cyclic nature of the domain is implicit in its own shape.

The coordinate centre of the domain is set at the centre of the toroid with the z axis pointing collinear to the revolution axis of the toroid. The toroid is set with a major radius of 0.8π and a minor radius of 0.33π.

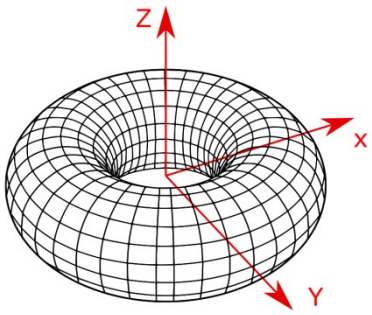

Figure 27: Toroidal domain coordinate scheme.

The meshing procedure starts by generating a block in blockMesh tessellated by hexahedrons. The block must be set to fully encase the toroid in order for the snappyHexMesh utility to tailor the base mesh to the toroid surface which is supplied as an STL file. Once the mesh is tailored to the toroid, the mesh inside fitted to the toroid surface by snappyHexMesh and cell layers are laid in on the inner surface of the toroid.

5.3.1 Fields set up

The first step towards generating any field in a toroidal domain is to determine mathematically the important toroidal values for each cell position as a function of cartesian coordinates. These “pseudo coordinates” and vectors are fundamental to compute the field values at each cell position:

The vector describing a cell centre coordinate (always referred to the coordinate centre of the mesh $(0,0,0)$) of a cell is denoted $\vec{p}$ and its unit vector $\hat{p}$. The angle α of the vector $\vec{p}$ to the vector $(1,0,0)$ projected on the xy plane can be obtained by calculating the inverse of the tangent for the (x, y) coordinates of the $\vec{p}$ vector accounting for the right angle sing. The code used to obtain the angle α from the (x, y) of $\vec{p}$ coordinates is shown here (as programmed in python syntax):

```
1.  def angleofvector2d(x, y):
2.      if (x == 0):
3.          if (y > 0):
4.              return (math.pi)/2
5.          else:
6.              return (math.pi)*1.5
7.      elif (y == 0):
8.          if (x>0):
9.              return 0;
10.         else:
11.             return math.pi;
12.     else:
13.         if (x>0 and y>0):
14.             return math.atan(y/x)
15.         elif (x<0 and y>0):
16.             return math.atan((-x)/(y))+math.pi/2
17.         elif (x<0 and y<0):
18.             return math.atan((-y)/(-x))+math.pi;
19.         else:
20.             return math.atan((x)/(-y))+math.pi*1.5;
```

The centre of the cross-section of the torus containing the cell of coordinates of $\vec{p}$ can be obtained as:

$$\vec{c} = \left(\mathrm{R} \cdot \hat{p}_x, R \cdot \hat{p}_y, 0\right). \tag{100}$$

where R is the major radius of the toroidal domain. The vector difference between $\vec{c}$ and $\vec{p}$, denoted $\vec{d}$ is effectively the vector describing the minor axis radius and orientation for the cross-section of the toroid where the cell is located and the cell’s coordinates.

$$\vec{d} = \vec{p} - \vec{c}. \tag{101}$$

The length of the $\vec{d}$ vector is defined as the minor radius of the toroid at cell’s centre coordinates and the unity vector is a pointer from the centre of the toroid cross-section at cell’s coordinates to the cell.

With these “pseudo coordinates” set the field generation is much easier to achieve.

In order to set the appropriate fields for the velocity and magnetic field, the following procedure has been used:

- Velocity field: The velocity field for the toroidal test case is equal to the field proposed for the cylindrical case. The plasma will be given a randomly generated velocity vector for each cell in the mesh in the range of o to 1e-6 m/s. The mathematical definition of the field is, given the cell coordinates (x, y, z)

$$\vec{u} = (u_x, u_y, u_z) = \left(1 \cdot 10^{-6} \cdot U(0,1), 1 \cdot 10^{-6} \cdot U(0,1),\ 1 \cdot 10^{-6} \cdot U(0,1)\right). \quad (102)$$

being $U(a, b)$ a continuous uniform distribution in the closed interval $[a, b]$.

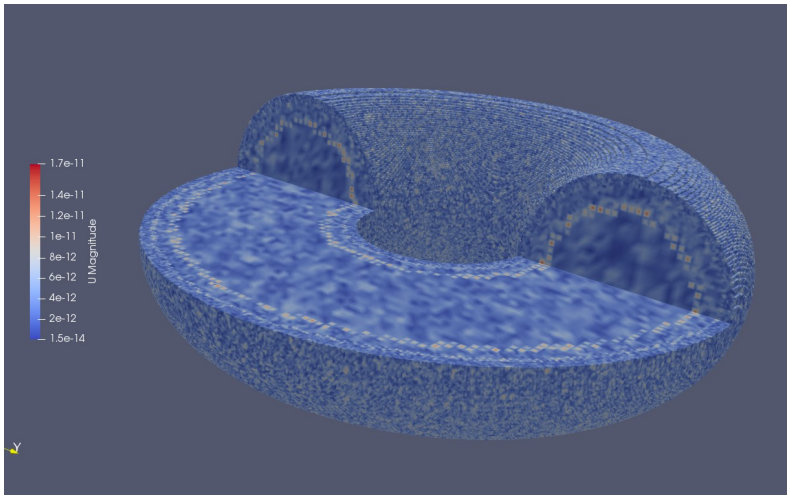

Figure 28: Initial velocity profile for the toroidal case (interpolated).

- Magnetic field: In order to generate the magnetic field configuration, firstly it should be mathematically defined.

  The imposed magnetic field ought to have an axial component which is perpendicular to the toroid cross-section and circulates through the toroid in clockwise direction. This component of the field is the toroidal field as described in previous chapters and can be computed from the previously defined “pseudo coordinates” as:

$$\overrightarrow{B_\theta} = B_{\theta 0} \cdot \frac{R'}{|\overrightarrow{p}|} \cdot \hat{\theta}. \quad (103)$$

with $B_{\theta 0}$ being the maximum allowed toroidal field which is achieved at the outermost surface of the toroid, R' being the major radius of the toroid plus the minor radius of the toroid and $\hat{\theta}$ being the unity vector pointing perpendicular to the toroid cross-section. The $\hat{\theta}$ unity vector can be easily obtained by a simple vector rotation on the xy plane as described by

$$\hat{\theta} = rotate(\hat{p}, \alpha). \tag{104}$$

And the rotation procedure of the vector is described in python code as:

```
1.  def rotatevector2d(x, y, alpha):
2.      return (x*math.cos(alpha)-y*math.sin(alpha),x*math.sin(alpha)+y*math.cos(alpha), 0)
```

The poloidal is defined in the toroidal geometry as a field contained in the torus cross-section which revolves around the centre of the cross-section. It can be mathematically depicted as:

$$\overrightarrow{B_{\phi}} = B_{\phi 0} \cdot \frac{\overrightarrow{|d|}}{a} \cdot \hat{\phi}. \tag{105}$$

being $B_{\phi 0}$ the maximum poloidal field magnitude which is obtained at the outer surface of the toroid, a as the minor radius of the toroid and $\hat{\phi}$ the unit vector pointing in the revolving direction of the poloidal field. The unit vector $\hat{\phi}$ can be determined by applying a 3D rotation matrix to rotate $\hat{d}$ around $\hat{\theta}$ 90 degrees clockwise. The calculus is performed by applying the following rotation matrix A

$$A = \begin{pmatrix} \hat{\theta}_x^2 & \hat{\theta}_x\hat{\theta}_y - \hat{\theta}_z & \hat{\theta}_x\hat{\theta}_y + \hat{\theta}_z \\ \hat{\theta}_y\hat{\theta}_x + \hat{\theta}_z & \hat{\theta}_y^2 & \hat{\theta}_y\hat{\theta}_z - \hat{\theta}_x \\ \hat{\theta}_x\hat{\theta}_z - \hat{\theta}_y & \hat{\theta}_y\hat{\theta}_z + \hat{\theta}_x & \hat{\theta}_z^2 \end{pmatrix}, \tag{106}$$

$$\hat{\phi} = A\,\hat{d}. \tag{107}$$

The magnetic field can be then be generated via a python script being fed the cells coordinates by the file 0/cellcenters. The obtained magnetic field can be observed on Figure 29 and Figure 30.

The python script used to generate the field can be examined for the reader comprehension:

```
    x = float(decomp[0])
          y = float(decomp[1])
          z = float(decomp[2])
          r = math.sqrt(math.pow(x,2)+math.pow(y,2))

          x_barret = x/(math.sqrt(math.pow(x,2)+math.pow(y,2)))
          y_barret = y/(math.sqrt(math.pow(x,2)+math.pow(y,2)))
          z_barret = 0

          direccio_barret = (x_barret, y_barret, z_barret)

          alpha = angleofvector2d(x,y);
          Bt_barret = rotatevector2d(0, 1, alpha);

          centre = (direccio_barret[0]*R, direccio_barret[1]*R, 0)
          difference = (x-centre[0], y-centre[1], z)
          innerlength = math.sqrt(math.pow(difference[0],2)+math.pow(difference[1],2)+math.pow(difference[2],2))

          Bp_barret = get3dRotation(difference, Bt_barret, math.pi/2)


          o.write("("+'%12.8f'%(bp*Bp_barret[0]*innerlength+bz*Bt_barret[0]*(1/r))+" "+'%12.8f'%(bp*Bp_barret[1]*innerlength+bz*Bt_barret[1]*(1/r))+" "+'%12.8f'%(bp*Bp_barret[2]*innerlength+bz*Bt_barret[2]*(1/r))+")"+"\n")


def rotatevector2d(x, y, alpha):
return (x*math.cos(alpha)-y*math.sin(alpha),x*math.sin(alpha)+y*math.cos(alpha), 0)

def angleofvector2d(x, y):
if (x == 0):
    if (y > 0):
        return (math.pi)/2
    else:
        return (math.pi)*1.5
elif (y == 0):
    if (x>0):
        return 0;
    else:
        return math.pi;
else:
    if (x>0 and y>0):
        return math.atan(y/x)
    elif (x<0 and y>0):
        return math.atan((-x)/(y))+math.pi/2
    elif (x<0 and y<0):
        return math.atan((-y)/(-x))+math.pi;
    else:
        return math.atan((x)/(-y))+math.pi*1.5;

def get3dRotation(inputvector,axis ,alpha):
vector_length = math.sqrt(math.pow(inputvector[0],2)+math.pow(inputvector[1],2)+math.pow(inputvector[2],2))
axis_vector_length = math.sqrt(math.pow(axis[0],2)+math.pow(axis[1],2)+math.pow(axis[2],2))

x_a = axis[0]/axis_vector_length
y_a = axis[1]/axis_vector_length
z_a = axis[2]/axis_vector_length

x_v = inputvector[0]/vector_length
y_v = inputvector[1]/vector_length
z_v = inputvector[2]/vector_length

vector = (x_a, y_a, z_a)

matrix = numpy.ones((3,3))
ivector = numpy.array([[x_v],[y_v],[z_v]])
matrix[0][0] = math.cos(alpha)+(math.pow(vector[0],2)*(1-math.cos(alpha)))
matrix[0][1] = vector[0]*vector[1]*(1-math.cos(alpha))-vector[2]*math.sin(alpha)
matrix[0][2] = vector[0]*vector[2]*(1-math.cos(alpha))+vector[1]*math.sin(alpha)

matrix[1][0] = vector[0]*vector[1]*(1-math.cos(alpha))+vector[2]*math.sin(alpha)
matrix[1][1] = math.cos(alpha)+(math.pow(vector[1],2)*(1-math.cos(alpha)))
matrix[1][2] = vector[1]*vector[2]*(1-math.cos(alpha))-vector[0]*math.sin(alpha)

matrix[2][0] = vector[0]*vector[2]*(1-math.cos(alpha))-vector[1]*math.sin(alpha)
matrix[2][1] = vector[2]*vector[1]*(1-math.cos(alpha))+vector[0]*math.sin(alpha)
matrix[2][2] = math.cos(alpha)+(math.pow(vector[2],2)*(1-math.cos(alpha)))
result = matrix.dot(ivector)
```

```
77.  return (result[0][0], result[1][0], result[2][0])
```

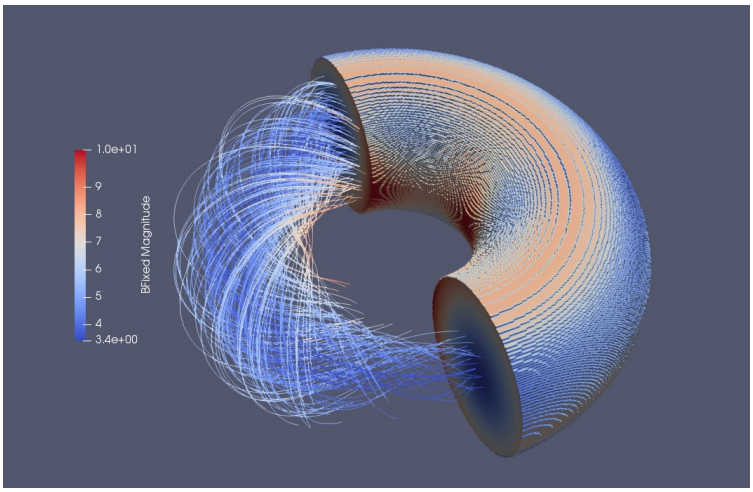

Figure 29: Externally imposed magnetic field configuration for the toroidal case.

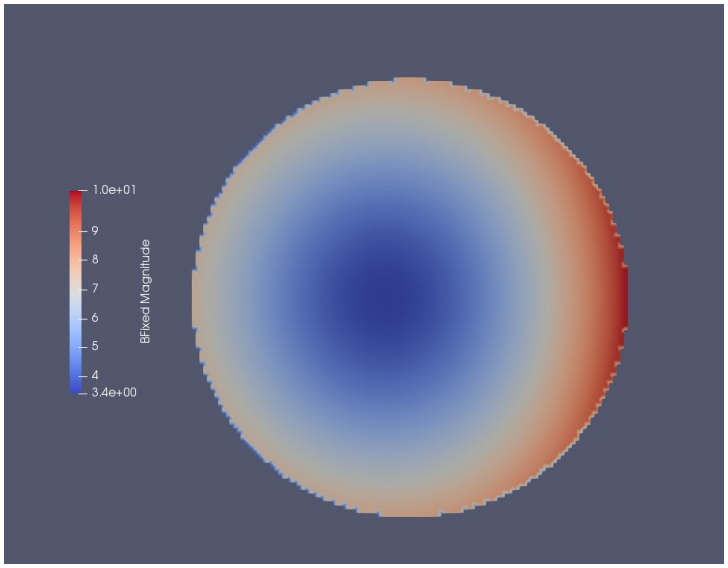

Figure 30: Externally imposed magnetic field for the toroidal case, cross-section.

5.4 Tokamak-like field configuration

Several scientific references on the fusion technology indicate that visco-resistive profiles on Tokamak-like fusion confinement machines have no constant profile. Therefore and aiming at providing experimental cases to be solved by the new modified OpenFOAM solver the visco-resistive field needs to be generated accordingly to a physically coherent profile. In the next subsection the visco-resistive field generation is discussed, but before proceeding to

generate such fields, a closer look can be taken into the expected Tokamak behaviour.

[10] indicates that a Tokamak fusion machine is to be expected to follow the given radial-dependent profiles as seen in Figure 32. It is expected to have a much higher pressure at the centre of the torus (taking as centre the coordinates where the minor radius becomes zero). The plasma pressure and temperature are expected to be considerably higher on that region and therefore the viscosity and resistivity are expected to have their minimum value on that local spatial region.

Current density also is expected to have a higher value on the central region of the toroidal domain as the lower resistivity local area enables a higher current density to flow through the centre.

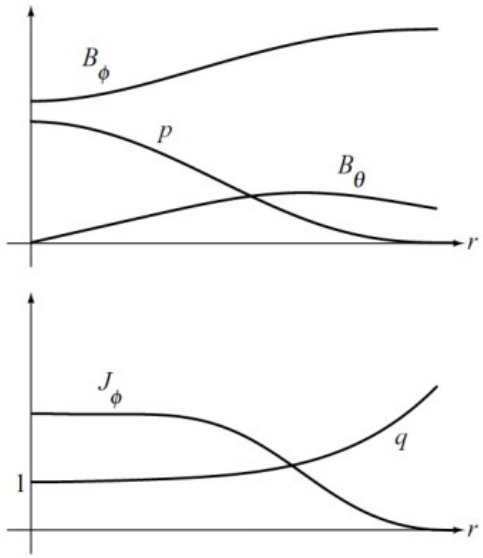

Figure 31: Typical profiles in a tokamak in the large-aspect-ratio limit [10].

In the cross-section of the toroidal domain, the plasma pressure is higher at the centre but also exhibits a slight drift towards the outer region of the Tokamak, this non-symmetrical patter as observed in Figure 32 is due to the non-magnetic symmetry on the field distribution as can be seen in Figure 30.

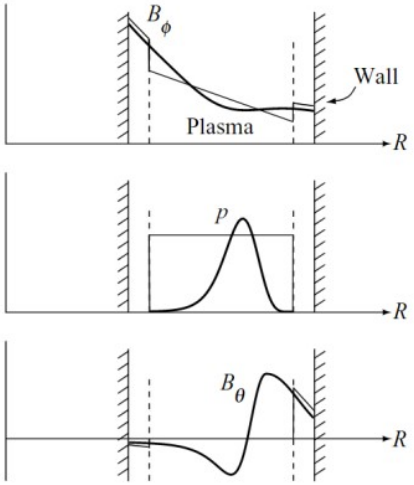

Figure 32: Typical midplane profiles (smooth curves) for the TFTR experiment. Superimposed is the surface current model approximation (box-like curves) [10].

5.5 Visco-resistive fields

The fields generated in order to test the solver for the test cases are set equally for both geometrical domains (cylindrical and toroidal) and consist in three possible radial-dependent profiles:

- Constant viscosity and resistivity: This is a benchmark field to test if the modified solver is fit to generate the same results as the original solver for a given uniform resistivity. If that field yields simulations with the same result as the original solver for the same resistive parameters we could ensure the solving and converging capabilities have not been lost purely due to the introduction of the new modified equation. The field in question can be implemented by using the procedure mentioned in the previous chapter.

- Linear profile of viscosity and resistivity: The field implemented for viscosity and resistivity has a radial profile ranging from η_{min} and υ_{min} to η_{max} and υ_{max} respectively for each field.

  The scalar field is calculated as a function of the minor radius in the toroidal geometry and as function of the radius for the cylinder (denoted r):

$$\nu(r) = \frac{r}{R}(\upsilon_{max} - \upsilon_{min}) + \upsilon_{min}, \tag{108}$$

$$\eta(r) = \frac{r}{R}(\eta_{max} - \eta_{min}) + \eta_{min}. \tag{109}$$

This field is used as a simple test to see the possible behaviour of the solution when a small variation into the visco-resistive field is introduced.

- Physical profile: This radial profile becomes much closer to the real gradients experimented in a magnetically confined plasma fusion machine. The definition of the visco-resistive profile is obtained from [9] as follow:

$$\eta(r) = \eta_0 \, f(r), \tag{110}$$

$$\nu(r) = \nu_0 \, f(r), \tag{111}$$

$$f(r) = \left(1 + \left(\frac{r}{a}\right)^{2\Lambda}\right)^{1+1/\Lambda}. \tag{112}$$

With Λ equal 4. This profile is much closer to the reality and can yield results comparable to external references to check for similitude in the results obtained.

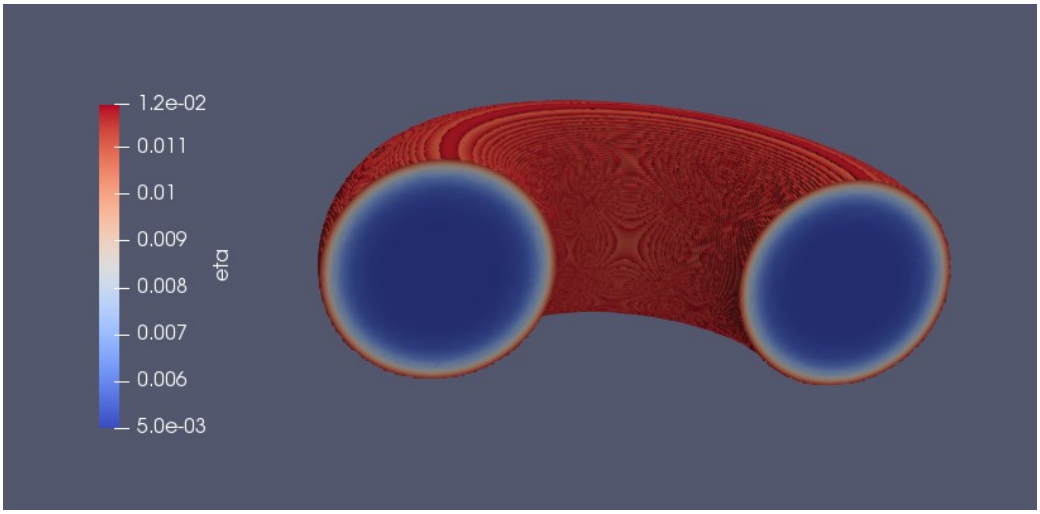

Figure 33: Resistivity profile for the toroidal case.

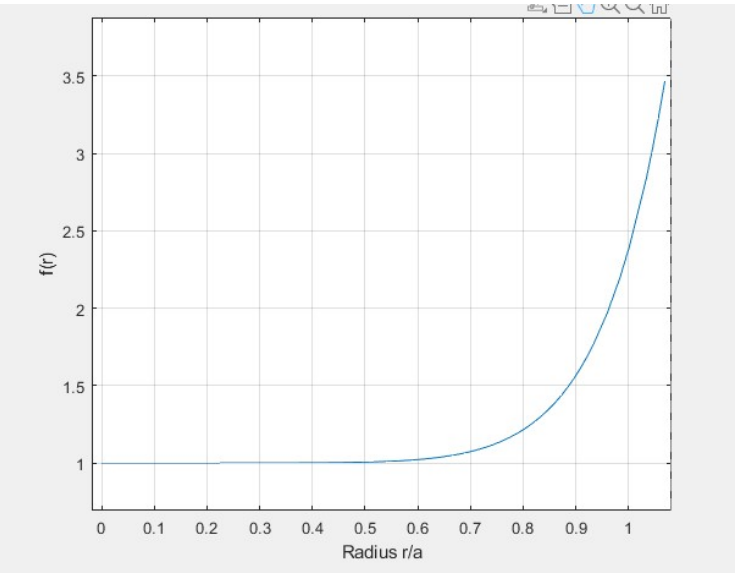

Figure 34: Visco-Resistive physical profile as obtained from [9].

Chapter 6: Results

6.1 Cylindrical domain results

Structured cylindrical meshing schemes have been found to consistently yield numerical errors and inaccuracy which resulted in incoherent results through the simulation. This effect was already observed in previous works [2] but was attributed to parallelization errors during the mesh decomposition step to prepare the case to be run in a multi-CPU system. The results obtained during this work indicate the mesh problem is not likely to be related to decomposition or mesh generation but rather to a solver-reading issue occurring when reading the case's mesh.

While the mesh generation, decomposition and later solution phases do not yield any error concerning the case's mesh there's a clear influence factor on how the solver interacts with the cells and their faces. It is worth mentioning that the simulations were performed in single CPU mode to ensure no parallelisation error was possible.

As can be seen the case solution during the first time intervals reveals a clear numerical pattern on the outer block's inter-block boundaries which does not correspond to any physical behaviour. The exhibited behaviour, which is clearly visible on the velocity field of the cylindrical domain for any magnetic field configuration (see Figure 35), appears to be caused by a lack of flux communication among cell faces on the inter-block boundary. It is worth mentioning the mesh generation procedure with blockMesh does join any adjacent block faces automatically as long as they share the same cell structure and distribution, therefore the inner-block faces are removed and cannot be assigned any boundary condition by the user or OpenFOAM.

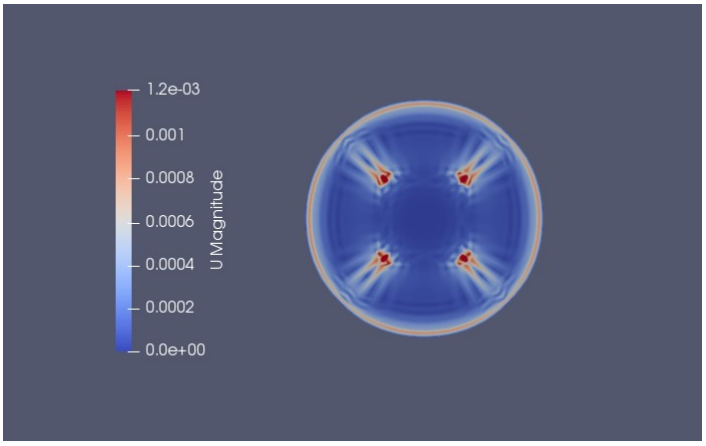

Figure 35: O-grid Mesh solution numerical noise. Velocity field obtained for a constant visco-resistive field and independent from the applied magnetic field. Result obtained at $t = 5s$.

This issue has only been detected on the cylindrical domain as it is the only domain generated by using a multi-block meshing geometry and previous works only performed analysis on geometries not involving this mesh generation method such as toroidal or cubic geometries [2].

Further attempts to correct the error by using axi-symmetric meshes have resulted in rather different results, as can be seen in Figure 36, where the axisymetric mesh yields a singularity-like velocity field. This result can indicate the mesh is not valid (as the O-grid mesh) not because inter-block flux communication but rather due to highly skewed cells, which are the ones located in the centre of the cylinder face and are the ones exhibiting a non-physical behaviour. Although the O-grid mesh does not contain highly skewed cells as the geometry is intentionally used to avoid that effect that could indicate the problem is related to inter-block cell communication and non-orthogonality; both the cells behaving strangely in the axi-symmetric mesh and the O-grid mesh are thought to have non-orthogonal faces on the block boundaries, which could be a possible reason for the behaviour.

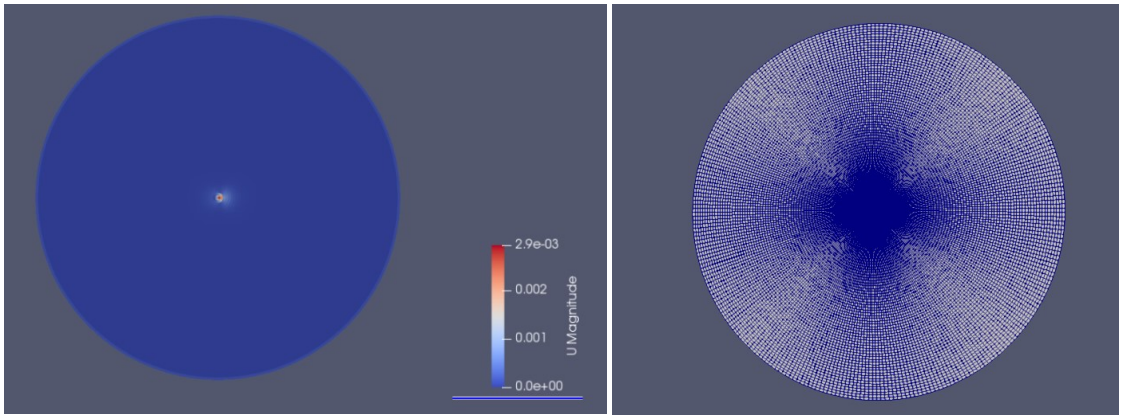

Figure 36: Comparison of the obtained results with the structured axi-symmetric mesh on the U velocity field at $t = 5s$ for a constant visco-resistive field of $(\eta, \nu) = (5 \cdot 10^{-2}, 5 \cdot 10^{-2})$ and independently from the applied magnetic field configuration.

Additionally, the unstructured cylindrical mesh obtained by using snappyHexMesh on a blockMesh based mesh was simulated in aim of finding a better performance by removing the inter-block boundaries. A snapped mesh has no blocks as the initial block is defined on blockMesh is unique and transformed into a single mesh with no block boundaries. Upon calculating the result of that mesh domain the results obtained kept being negative in physical plasma behaviour. Nonetheless, the troubled cells were located at the outer surface of the mesh rather than the inner area as observed with the other meshes.

These results indicated the problem was more likely related to the deformed cells on the boundaries between blocks and surfaces of the geometrical domain. Although these results do not yield any definitive conclusion on what the exact problem is, an approach of how to solve it was taken by using unstructured meshes generated by automatic algorithms which ensured no highly skewed or non-orthogonal cells and faces were produced; such an algorithm is the Gmsh algorithm provided in the Gmsh software package.

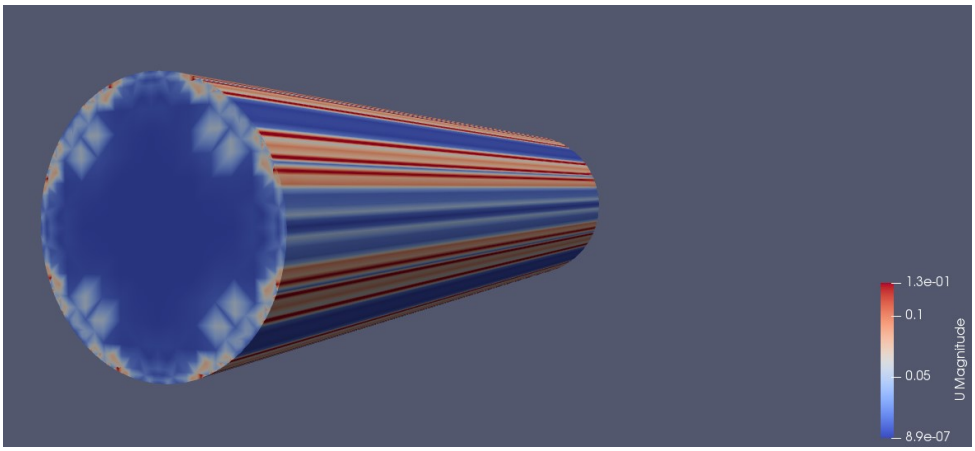

Figure 37: Faulty results obtained by using the snapped mesh on the cylindrical domain. Velocity field obtained at $(t = 5s)$ for a constant visco-resistive field of $(\eta, \nu) = (5 \cdot 10^{-2}, 5 \cdot 10^{-2})$ and independently from the applied magnetic field configuration.

The unstructured cylindrical mesh generated using Gmsh yielded coherent simulation results without any visible numerical pattern as the ones previously detected. Interesting results showing a helical pattern on the Z component of the velocity field have been found and are considered promising results due to their resemblance to the results obtained by [24] (see Figure 38, 39 and 41).

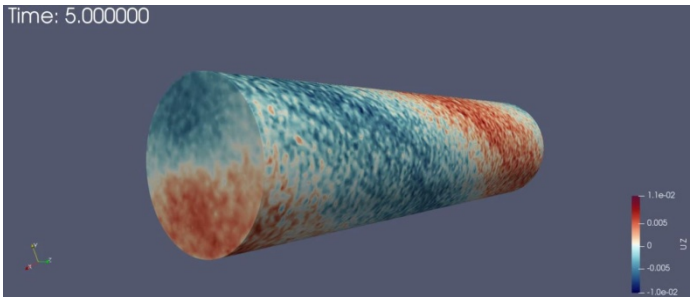

Figure 38: Z component of the velocity field obtained on the unstructured cylindrical mesh for a pinch ratio of 1.56, $(B_0, B_c) = (4.5, 7.06)$ and constant resistivity and viscosity fields $(\eta, \nu) = (5 \cdot 10^{-2}, 5 \cdot 10^{-2})$.

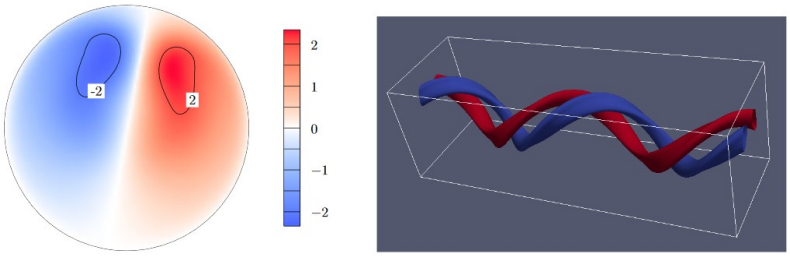

Figure 39: Results obtained by [24] showing the helical self-organization of plasma on the cylindrical geometry.

In the simulation performed with the unstructured cylindrical mesh the helical behaviour of the plasma showed in the Figure 38 fades over time while sudden polarity changes occur in the z component of the velocity field. The fading phenomena of the plasma behaviour are attributed to the resistivity imposed on the simulation which as a dissipative process causes a loss in energy on the system and thus decreasing the velocity magnitude on the overall system.

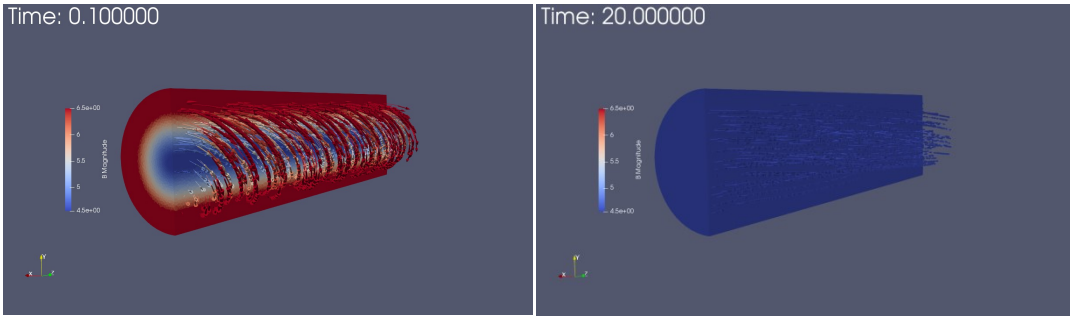

Figure 40: B field during the unstructured cylindrical mesh solving procedure (at $t = 0s$ (left) $and\ t = 20s$ (right)).

Figures 40 and 42 expose the inducted magnetic field phenomena. On Figure 40 the magnetic field induced in the plasma is plotted both as a colour profile of $|\vec{B}|$ and a vectorial plot of $\vec{B}$. The result of the comparison of the first and last time steps of the simulation is clear, the field induced in the plasma counteracts the poloidal magnetic field component and opposes to the external magnetic field, making the total B field to lack of poloidal magnetic field at the end of the simulation. Figure 42 exposes the evolution of the B_{bvle} magnetic field, which only has poloidal component and counteracts the magnetic field imposed externally.

Further analysis of the results on the unstructured cylindrical mesh solution showed a fading magnetic field on the poloidal direction but not on the toroidal direction. These phenomena need to be studied in further works to ensure if it is the result of incoherent mhdFoam solver solving procedure, a meshing issue affecting the solver solution or rather it is an expected result due to the imposed magnetic field.

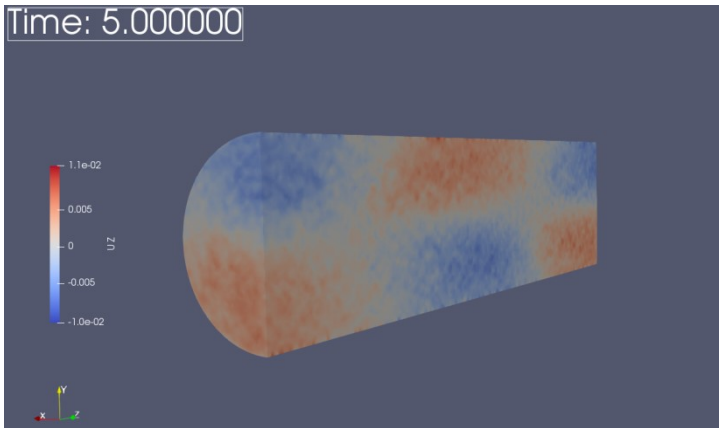

Figure 41: Detail of a cross-section of the cylindrical unstructured mesh velocity field on the Z direction. The helical pattern can be observed in the inside of the plasma volume. The magnetic and visco-resistive configuration is the same as on Figure 38 at the same time step.

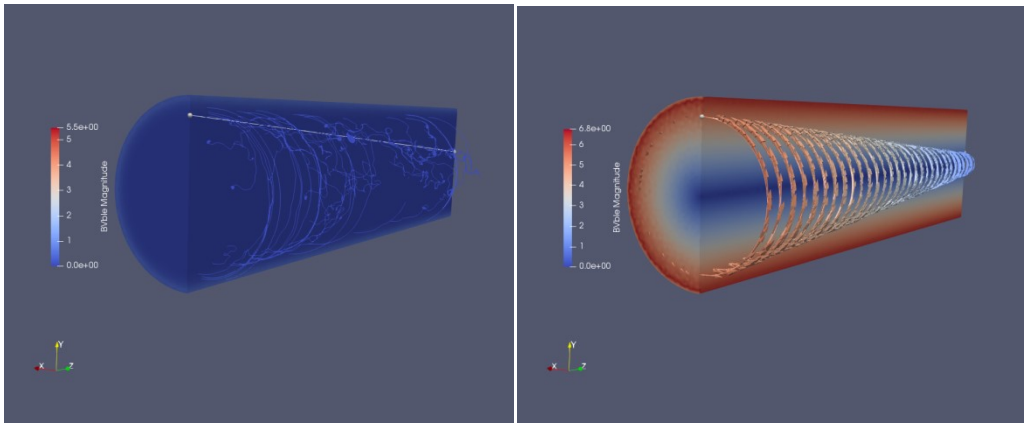

Figure 42: Detail of the evolution of the magnetic Bbvle field on the unstructured cylindrical mesh at time $t = 0s$ (left) and $t = 20s$ (right).

Figure 43 shows the evolution of Kinetic and magnetic energy contained on the cylindrical mesh during simulation both for the overall magnetic field and also the variable magnetic field. The results show a clearly dissipative process being present during the simulation as the kinetic energy clearly decreases. On the first simulation time steps the velocity field suddenly increases as the external magnetic field is applied. As the velocity field acquires its helical structure the field is weakened by the constant resistivity effect which is naturally a dissipative process.

The total magnetic field B is weakened from the first time step magnetic field (which is the magnetic field imposed externally). The magnetic field energy associated to the variable magnetic field increases as the plasma opposes to the external magnetic field poloidal component (this effect should be further researched in future works to yield an explanation to

these phenomena) until the magnetic field reaches an equilibrium with the external magnetic field and the induced field on the plasma only consists of the external magnetic field axial component.

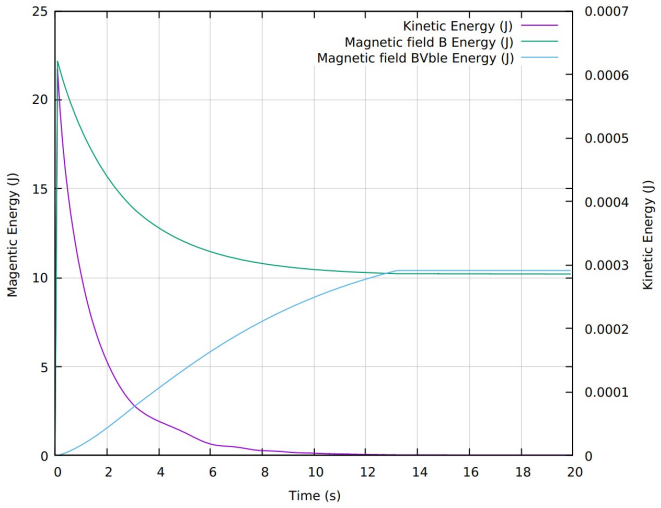

Figure 43: Kinetic and Magnetic energy plots during the cylindrical geometry simulation.

Also the residuals from the cylindrical simulation have been analyzed in order to find possible problems leading to solution convergence issues. In Figure 44 time step delta of the simulation is plotted against the maximum Courant number and its mean value across the mesh. The simulations have been performed with adjustable time step to ensure CFL conditions are met within the procedure. OpenFOAM analyzes the Courant Number on every simulation step and compares it to a predefined threshold of 0.5. If the maximum Courant number is over 1 the simulation can yield results leading to non-convergence and thus the results would not be valid.

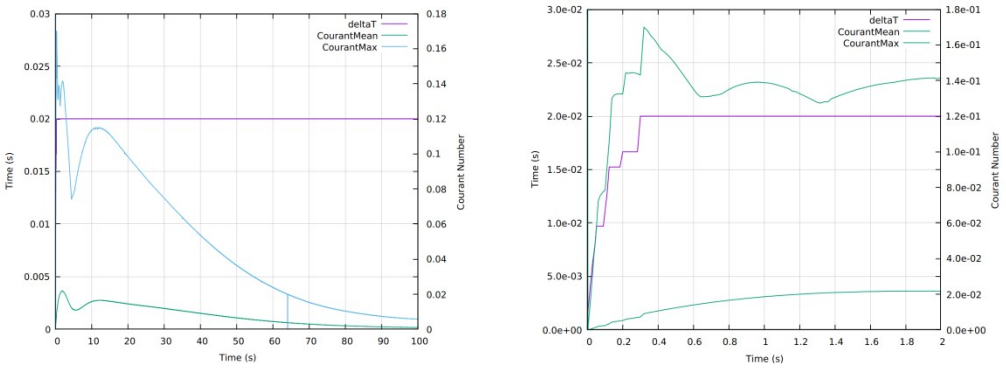

Figure 44: Courant number and simulation timestep evolution during the simulation procedure of the cylindrical mesh (left) and the detail of the first time steps of the simulation (right).

As can be observed, the timestep of the simulation is greatly reduced at the beginning during the first time steps but later rises rapidly to the maximum allowed value of 0.2 s. As can be seen the Courant number decreases constantly during the simulation denoting a valid CFL condition.

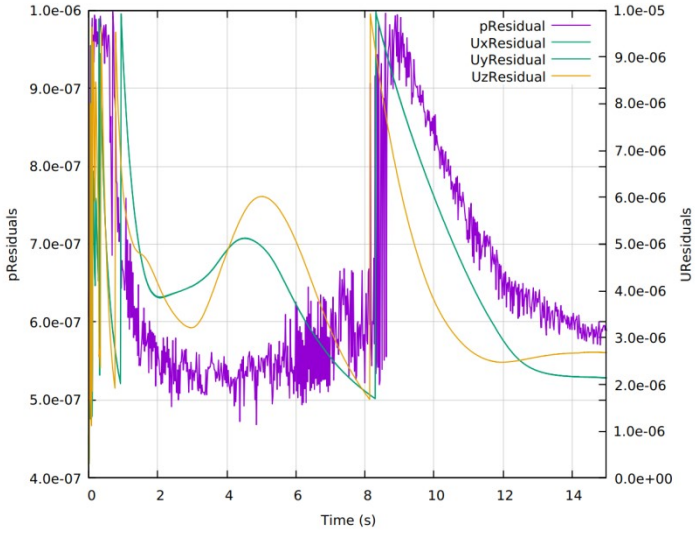

Figure 45: Velocity equation residuals during the simulation for the cylindrical mesh.

As it is shown in Figure 45 the residuals obtained during the solving process for the velocity equation model are not converging and oscillating instead. This is thought to be caused by the simulation not reaching the steady-state as the pulsating effect of the velocity field causes the simulation to switch the velocity field direction periodically until the kinetic energy is reduced to the point no pattern can be observed and the simulation reaches steady state by extinguishing the velocity completely.

6.2 Toroidal meshing results

In the toroidal domain the mesh generated as described previously proved to be solvable and yield physical results as expected. On previous works on this mesh and the mhdFoam solver [2] the mesh yielded similar results.

The initial velocity field, which is configured as a constant evolves into the visible patterns on Figure 46, which exposes the velocity field on a cross-section of different time steps on a toroidal geometry. The system shows clear energy dissipation due to the constant resistivity and viscosity. The pattern obtained shows a clearly visible plasma circulation inside the toroidal geometry.

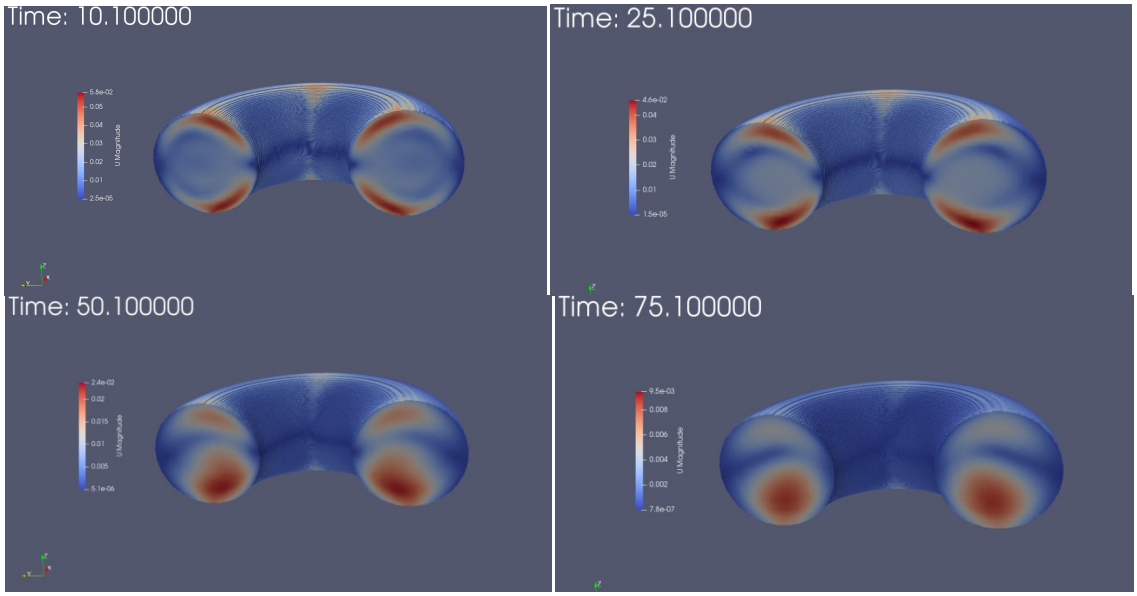

Figure 46 : Velocity field magnitude temporal evolution of the toroidal mesh with constant resistivity and viscosity of $\eta = \nu = 5e-2$ and field configuration of $B_\theta = B_\phi = 0.7$ with a pinch ratio of 1.

Further tests have been carried out with the same mesh by generating a parameter scan of the pinch ratio and the visco-resistive field. Firstly the same toroidal mesh is computed for three pinch ratios of the magnetic external field, 0.5, 1 and 2, making the toroidal field constant at 0.7 and changing the poloidal field from 0.35 to 1.4. These simulations were performed with a constant resistivity and viscosity field of $\eta = \nu = 5e-2$. The same procedure was repeated with a viscous and resistive field shaped by the profile mentioned in the previous chapter,

$$\eta(r) = \eta_0 \, f(r), \tag{113}$$

$$\nu(r) = \nu_0 \, f(r), \tag{114}$$

$$f(r) = \left(1 + \left(\frac{r}{a}\right)^{2\Lambda}\right)^{1+1/\Lambda}. \tag{115}$$

by setting $\eta_0 = \nu_0 = 5e-2$, $a = 0.33\,\pi$ and $\Lambda = 4$.

The results of the parameter scan were computed for the kinetic and magnetic energy, computed for the whole mesh at each individual time step

Also the calculation of the Reynolds magnetic number and the Hartmann number were thought to be important for the analysis, so they're computed and graphed as well. In Figure 47 and Figure 48 the simulation results for a magnetic configuration of $B_\phi = B_\theta = 0.7$ and constant visco-resistive field vs. non-uniform visco-resistive plasma field can be appreciated. Both figure illustrate the behaviour of the plasma upon reaching steady-state (the selected time step is *t = 75s*).

The first magnitude graphed across the toroidal cross-section is the Magnetic Reynolds number which is a physical non-dimensional parameter of an MHD plasma expressing the ratio of inductive to diffusive phenomena on the plasma behaviour as denoted by:

$$R_m = \frac{UL}{\eta}. \tag{116}$$

where R_m is the Reynolds magnetic number, U is the velocity field of the plasma, L is a characteristic longitude taken here as 1 and η being the resistivity of the plasma. A higher magnetic Reynolds number expresses weaker diffusive phenomena on the plasma.

A clear difference can be appreciated in Figure 47 on the edges of the toroidal cross-section, where the resistivity increases in the non-uniform case (bottom) and a lower Reynolds number is observed. Therefore, a stronger diffusion phenomenon-is observed at the wall sections of the toroidal volume in terms of the magnetic Reynolds number. This result is logical due to the higher resistivity near the walls of the toroid and therefore a lower magnetic Reynolds is expected.

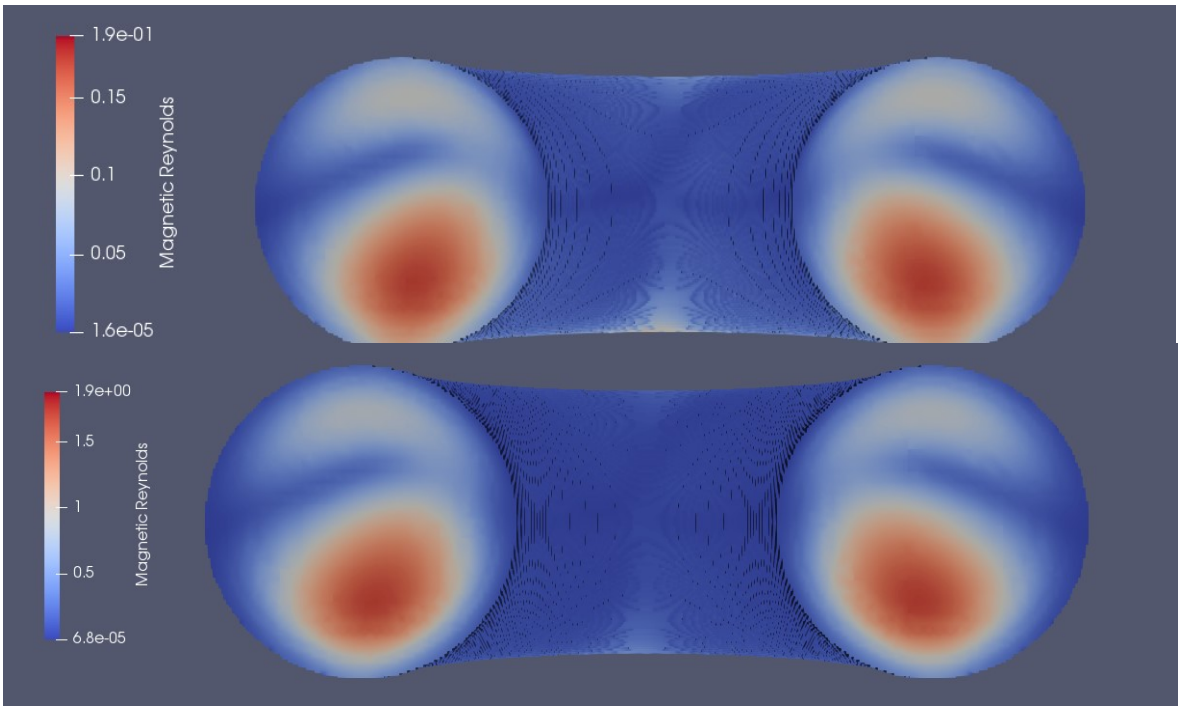

Figure 47: Magnetic Reynolds profile for the toroidal cross-section on a uniformly resistive plasma (top) with $\eta = \nu = 5e-2$ and a plasma with a visco-resistive distribution as denoted by the visco-resistive profile exposed on Chapter 5 with $\eta_0 = \nu_0 = 5e-2$. Both cases were simulated with identical initial magnetic and velocity fields.

The other parameter to be studied on the yielded simulation is the Hartmann number, which is a common parameter on MHD that denotes the ratio of the magnetic to viscous forces which the plasma experiences. It can be mathematically denoted as:

$$Ha = BL\sqrt{\frac{1}{\nu\eta}}. \tag{117}$$

where Ha is the Hartmann number, B is the magnetic field experienced by the plasma, L is a characteristic length which will be considered as 1 and ν, η being viscosity and resistivity respectively.

Figure 48 shows the Hartmann number profile on a cross-section of the simulated toroidal case as explained previously. The Hartmann number slightly weakens towards the wall surface of the toroidal cross-section as a consequence of a higher resistivity and viscosity near the wall. The difference can be observed on the top and bottom images of Figure 48.

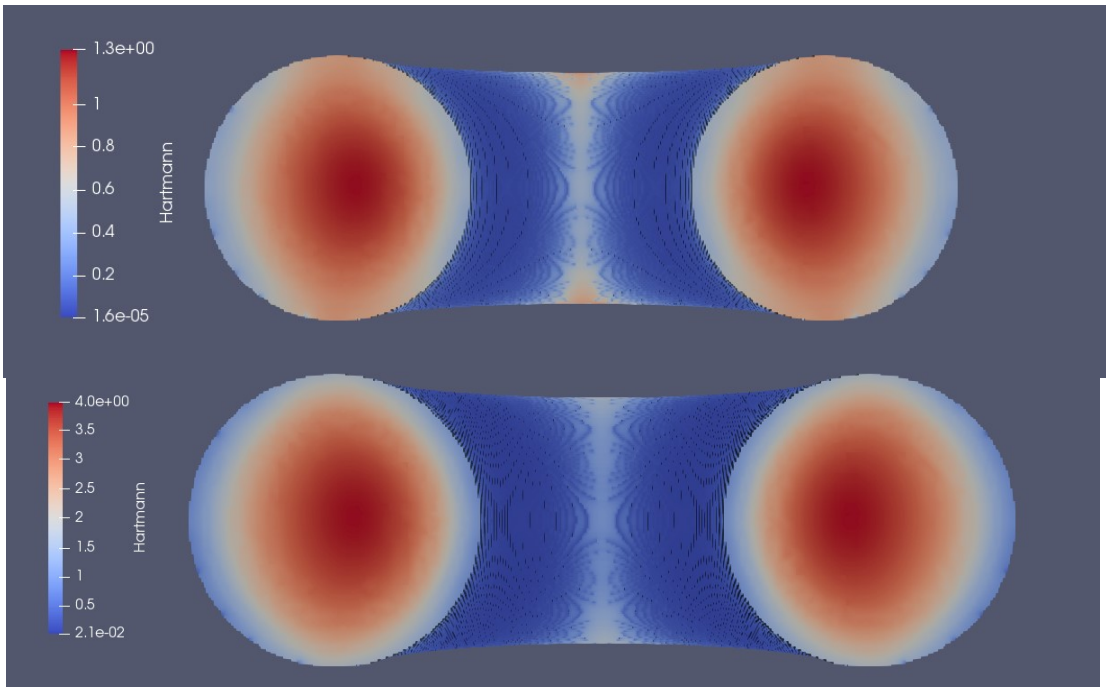

Figure 48: Hartmann number profile for the toroidal cross-section on a uniformly resistive plasma (top) with $\eta = \nu = 5e-2$ and a plasma with a visco-resistive distribution as denoted by the visco-resistive profile exposed on Chapter 5 with $\eta_0 = \nu_0 = 5e-2$. Both cases were simulated with identical initial magnetic and velocity fields.

The parameter profiles on the toroidal cross-section also are affected by the difference in viscosity and resistivity as can be seen in Figure 49. It is worth noting the current density vector has a clear minimum in the centre of the cross-section, meaning that while plasma is moving on the central area of the section the current charge is being developed on the extremes of the section.

Magnetic Reynolds number and Hartmann number are also seen to describe a profile with a clear maximum at the centre where the velocity of the plasma is higher and the magnetic field is also more intense. The profiles shown in Figure 49, which correspond to a toroidal mesh case with the previously explained conditions of non-uniform visco-resistive fields, show a physical behaviour of the plasma slightly different on the *U* and *J* field curves compared to the cases solved with uniform visco-resistive fields as can be seen in Figure 50 and 51.

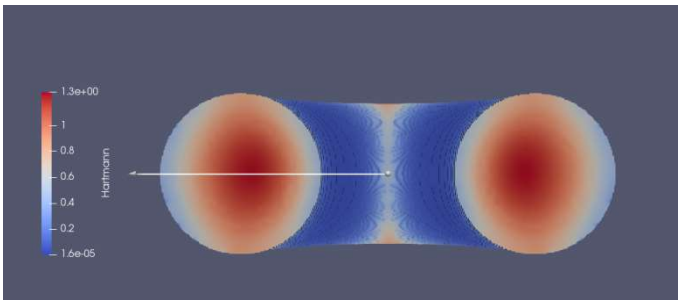

Figure 49: Cross-section of the toroidal domain showing along which line are the parameters plotted on Figure 50 and 51.

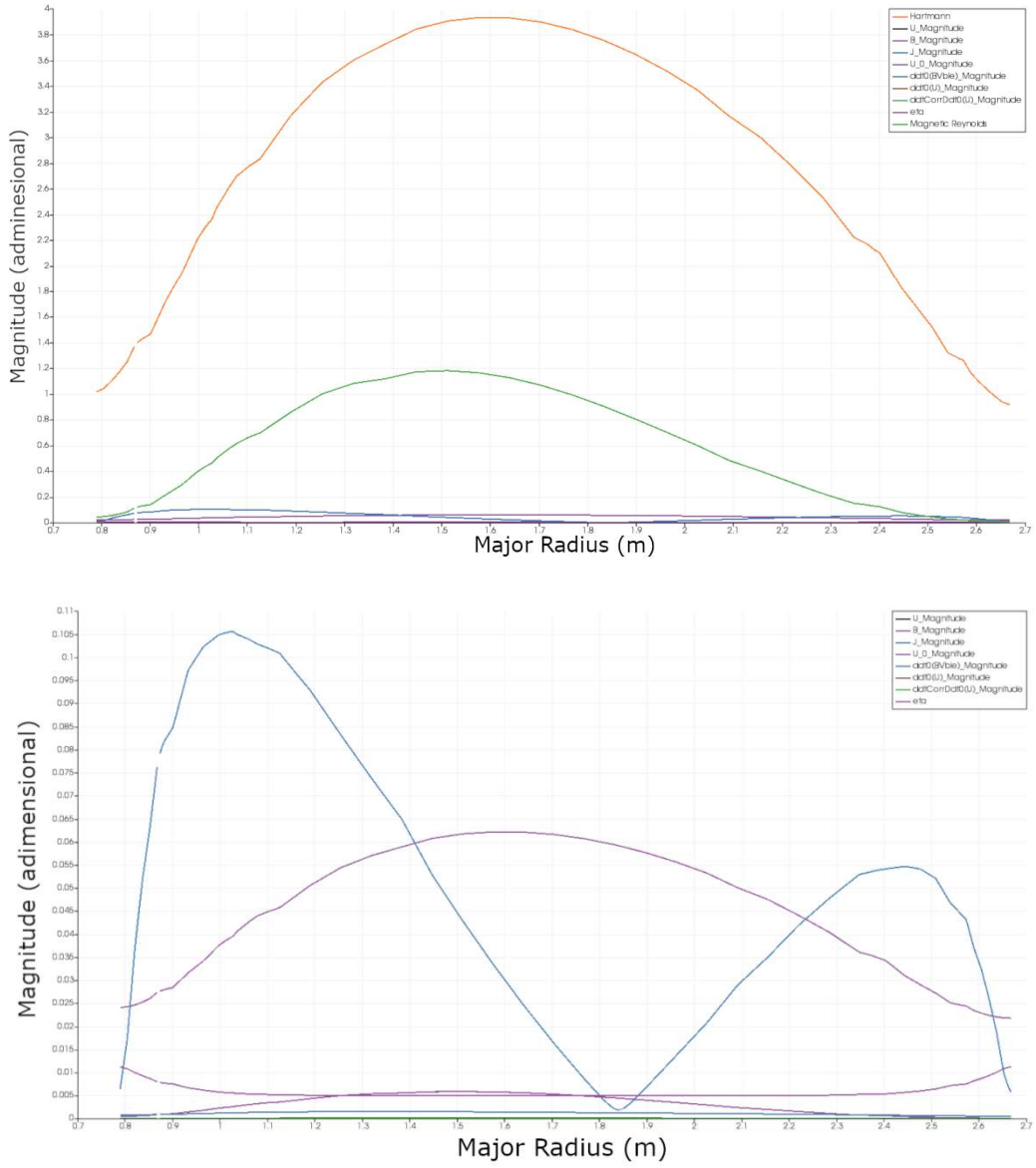

Figure 50: Parameter profiles for a non-uniform visco-resistive plasma case with $\eta_0 = \nu_0 = 1e-2$ and a magnetic field configuration of $B_\phi = B_\theta = 0.7$.

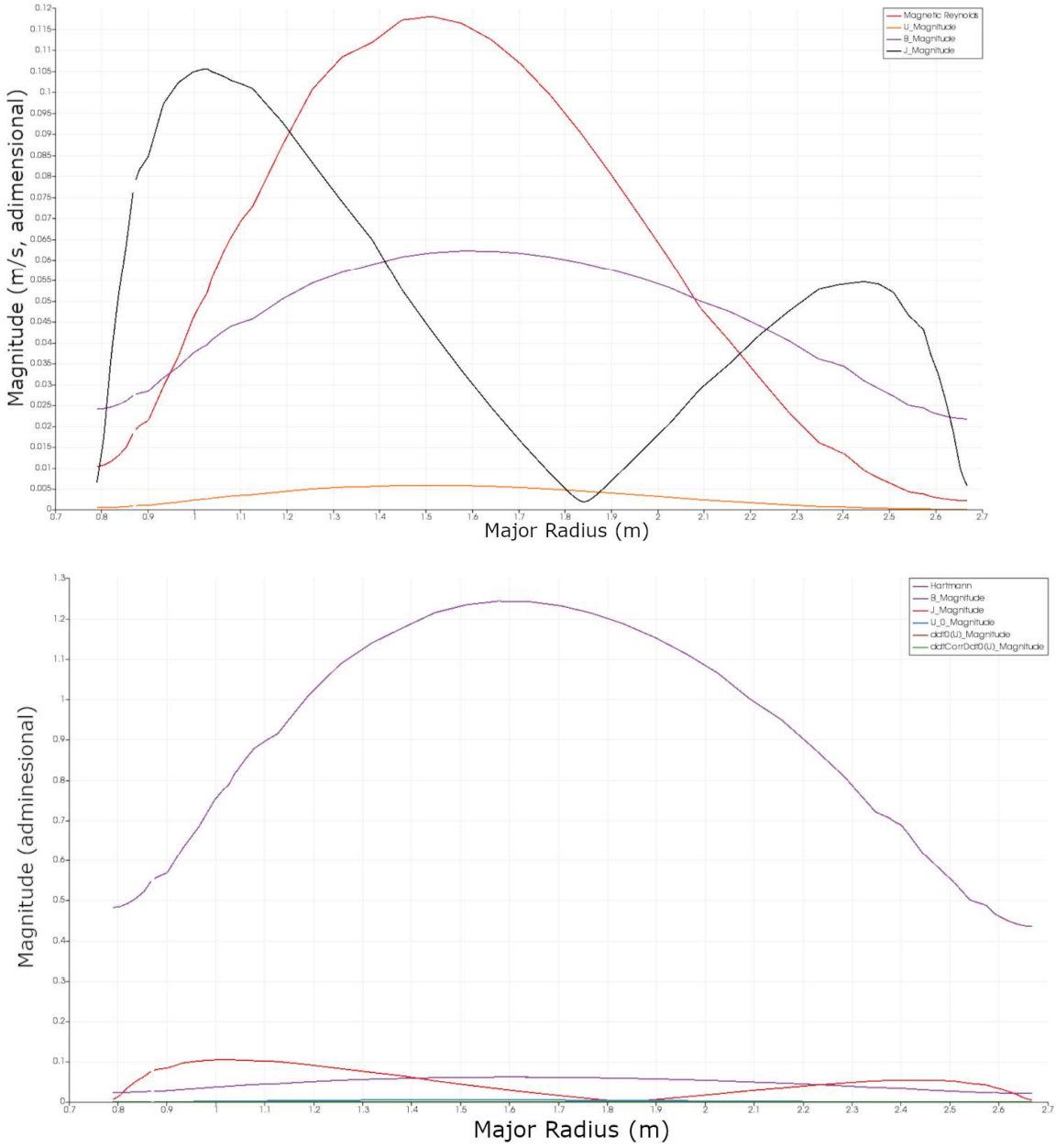

Figure 51: Parameter profiles for a uniform visco-resistive field applied on a toroidal meshed case with magnetic configuration of $B_\phi = B_\theta = 0.7$.

The results obtained in this work do not completely prove performance of the new code as better test cases should be proposed and solver in order to obtain complete confirmation of the solver performance. The toroidal and the cylindrical meshes both posed some level of issues while simulating due to meshing issues related to resolution and/or mesh structure; during the work the meshes have been improved in order to obtain some results and try to, at

least. Prove the solver can compute a valid solution without magnetic divergence and yield some physical results.

Future work should aim at providing a proper validation and benchmark of the modified code as done in [2]. Given the results obtained the solver is thought to yield promising results, but a benchmark is needed to prove that assumption.

It is worth mentioning that some resistivity, viscosity and external magnetic fields produced a highly diverging magnetic flux value, making the solution impossible to achieve thorough the simulation. Such cases were discarded as they were configured with excessively large magnetic fields which caused magnetic flux to diverge.

Some simulations also yielded problematic results as the configuration made them unsolvable by applying correct CFL conditions. Such cases presented a time delta between simulation steps which always decreased until OpenFOAM crashed due to excessively low values which could not be fitted in a float type value. These cases were automatically discarded due to having inadequate meshes which triggered high Courant numbers and forced the OpenFOAM adjustable time step to constantly decrease; such effects are shown in Figure 52.

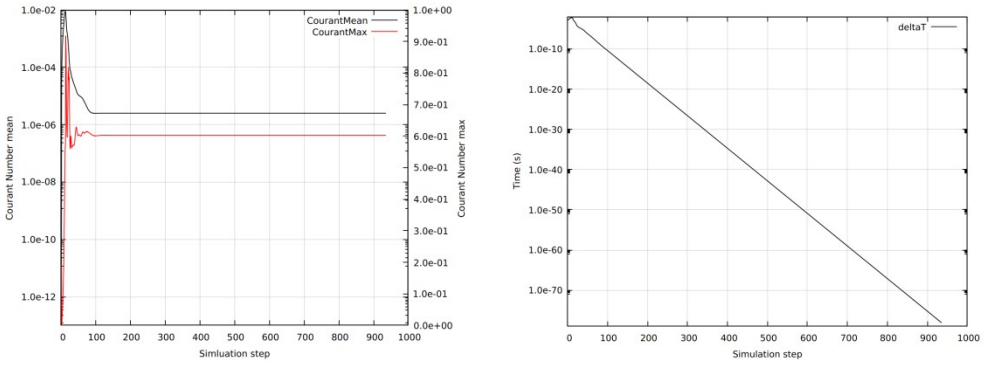

Figure 52: Courant number conditions obtained in wrongly meshed toroidal cases. Courant Number maximum and mean values (left) are over the desired thresholds causing the simulation to automatically decrease its delta between the time steps (right).

Chapter 7: Economical and environmental study.

In this chapter a brief explanatory note will be given to summarize the environmental footprint the present work has posed during its development as well as a detailed description of the costs of the project.

Environmental impact

As the project has consisted on a wide research on fluid mechanics and electromagnetics which resulted in the usage of CFD simulations the only environmental impact to be considered is that of the computers used for numerical analysis.

The computer cluster used on this project to generate and run the simulations is the MARCONI supercomputer at the CINECA consortium facility. The carbon footprint of the computer is not detailed by CINECA, but can be inferred by using Lenovo's (the manufacturer of the computer nodes at Marconi) data of power consumption.

The Marconi supercomputer is dived into nodes, each of each containing 48 CPUs. Estimating a node power consumption of 300W (Source: Intel datasheets) from the CPU manufacturer (Intel) datasheet, the total energy consumption throughout the project development is estimated as:

$$E = P_n N_n t \cdot 1.4. \tag{118}$$

with P_n being the node power consumption of 300W, N_n being the mean number of nodes used during each simulation and *t* being the total simulation time consumed. A scaling factor of *40%* is applied as estimation for other power consumption such as network, data storage, cooling and login node systems. This is value represents an approximation to the real energy footprint.

Considering a total simulation time of 200 hours per node and that the simulations were run with 4 nodes each (192 CPUs) the total energy consumption equates to 336 KWh of electrical energy. Given a CO2 emission of 256g/KWh (Source: European Environment Agency) in Italy (the location of the supercomputer) the total CO2 emission is considered as 86.01 Kg of carbon dioxide.

Economical study

The economical cost of the study is considered as the cost of the hours of research work (the author), the cost of the supervisor work, the cost of supercomputer usage and other costs such as references access. The total study of the project is detailed in the table below.

The supercomputing power cost has been considered as the energy cost of the system (considering a price of 0.24€/KWh as stated by the European Environmental Agency) plus a 50% of the cost due to management and license cost.

Concept	Cost per hour (€/h)	Total hours (h)	Total cost (€)
Researcher work	30	400	12000
Supervisor work	50	30	1500
Supercomputing time	0.6	200	120
Reference access	-	-	100

Table 1: Estimated costs for the current work.

The software used to carry out this project poses no additional cost as only Open-Source free software tools have been used (OpenFOAM, Paraview, gnuplot and python).

The total economical analysis of the project yields an estimated cost of 13720€ for the present work.

Chapter 8: Conclusions

The present work contributes to the MHD physics and computational fluid dynamics by the usage of OpenFOAM C++ library. The aim of the work has been successfully achieved by providing a detailed guide on how the mhdFoam solver ought to be modified to account for non-uniform resistivity and viscosity thus improving its physical solving capabilities.

An OpenFOAM introduction has been proposed, so the reader can get familiar with the CFD package and be able to follow and understand the mechanics of the code improvement and the Numerics behind the solver. Special insight has been given to the numerical methods used to solve in particular the MHD model in the mhdFoam solver, which proved to be very useful when improving the original solver. This work contributes and encourages to junior researchers/engineers to open the door for the scientific quest of fusion research.

The code improvement has been performed step by step and justified in order to give the reader every detail of the procedure. The solver core code was slightly modified to account for non-uniform resistive and viscous fields and the different available solution schemes were discussed for the reader to understand the importance of the solution numerical schemes in the solving process.

The pre-processing of the plasma cases has been done by using different meshing tools such as Gmsh, blockMesh and snappyHexMesh and a variety of meshing procedures have been implemented for the cylindrical geometrical domain mesh, which proved to be an issue in previous OpenFOAM works [2]. The toroidal mesh was also produced with meshing techniques to achieve a computationally useful mesh. Additionally, a field generation procedure has been demonstrated to effectively generate the physical coherent magnetic and velocity fields for the proposed simulations on their respective geometrical domain. Lastly the non-uniform visco-resistive fields were demonstrated.

The results of the work proved to be useful in the aim of the project by providing data obtained through the solution of the proposed cases. Firstly the cylindrical meshing problem has been analyzed and a possible solution by using unstructured tetrahedrical meshes has been applied to solve the geometrical domain. Also, the generated toroidal mesh has proven to be of interest and use in obtaining results with the modified solver. The data obtained by using simulations proved to be useful in determining the capabilities of the solver, although they did not demonstrate completely the physical behaviour of the modified solver probably due to lack of mesh resolution and time to completely analyze the results. Further work needs to be done on the mesh and case analysis to prove the capabilities of the proposed modification to the solver to their full extent.

In future works the modified solver ought to be further tested in various geometrical domains and under several testing conditions in order to accurately prove that indeed the modified solver does yield physical results which are considered physically valid.

References

[1] OpenCFD, *OpenFOAM User Guide*, v2012 ed. 2020.

[2] D. Garrido González, "Non-linear MHD simulations of magnetically confined plasma using OpenFOAM," Bacherlor Thesis, Universidade de santiago de compostela, 2020.

[3] S. Futatani, S. Benkadda, and D. del-Castillo-Negrete, "Spatiotemporal multiscaling analysis of impurity transport in plasma turbulence using proper orthogonal decomposition," *Phys. Plasmas*, vol. 16, no. 4, p. 042506, Apr. 2009, doi: 10.1063/1.3095865.

[4] S. Futatani, S. Benkadda, Y. Nakamura, and K. Kondo, "Multiscaling Dynamics of Impurity Transport in Drift-Wave Turbulence," *Phys. Rev. Lett.*, vol. 100, no. 2, p. 025005, Jan. 2008, doi: 10.1103/PhysRevLett.100.025005.

[5] S. Futatani, S. Benkadda, Y. Nakamura, and K. Kondo, "Characterization of intermittency of impurity turbulent transport in tokamak edge plasmas," *Phys. Plasmas*, vol. 15, no. 7, p. 072506, Jul. 2008, doi: 10.1063/1.2947027.

[6] S. Futatani, X. Garbet, S. Benkadda, and N. Dubuit, "Impurity dynamics in the presence of transport barriers in tokamaks," *Phys. Plasmas*, vol. 17, no. 10, p. 102501, Oct. 2010, doi: 10.1063/1.3481462.

[7] S. Futatani, X. Garbet, S. Benkadda, and N. Dubuit, "Reversal of Impurity Pinch Velocity in Tokamaks Plasma with a Reversed Magnetic Shear Configuration," *Phys. Rev. Lett.*, vol. 104, no. 1, p. 015003, Jan. 2010, doi: 10.1103/PhysRevLett.104.015003.

[8] S. Futatani, D. del-Castillo-Negrete, X. Garbet, S. Benkadda, and N. Dubuit, "Self-Consistent Dynamics of Impurities in Magnetically Confined Plasmas: Turbulence Intermittency and Nondiffusive Transport," *Phys. Rev. Lett.*, vol. 109, no. 18, p. 185005, Nov. 2012, doi: 10.1103/PhysRevLett.109.185005.

[9] S. Futatani, J. A. Morales, and W. J. T. Bos, "Dynamic equilibria and magnetohydrodynamic instabilities in toroidal plasmas with non-uniform transport coefficients," *Phys. Plasmas*, vol. 22, no. 5, p. 052503, 2015, doi: 10.1063/1.4919960.

[10] J. P. Freidberg, *Plasma physics and fusion energy,* 2007.

[11] H. Klus, "Nuclear Physics." [Online], Available: http://www.thestargarden.co.uk/Nuclear-physics.html.

[12] Antunes R., "TRITIUM: A CHALLENGING FUEL FOR FUSION," *EUROfusion webpage*, Aug. 11, 2017. [Online], Available at : https://www.euro-fusion.org/news/2017-3/tritium-a-challenging-fuel-for-fusion/.

[13] C. Xu *et al.*, "Ramp-Up-Phase Current-Profile Control of Tokamak Plasmas via Nonlinear Programming," *IEEE Trans. Plasma Sci.*, vol. 38, no. 2, pp. 163-173, Feb. 2010, doi: 10.1109/TPS.2009.2037626.

[14] F. F. Chen, *Introduction to Plasma Physics and Controlled Fusion*. 2016.

[15] D. D. Schnack, *Lectures in Magnetohydrodynamics*, vol. 780. 2009.

[16] Wolf dynamics, "Finite Volume Method: A Crash introduction.", [Online] Available:

http://www.wolfdynamics.com/wiki/fvm_crash_intro.pdf .

[17] E. Mas de les Valls Ortiz, “Development of a simulation tool for MHD flows under nuclear fusion conditions,” PhD Thesis, UPC, 2011.

[18] J. . Brackbill and D. . Barnes, “The Effect of Nonzero ∇ · B on the numerical solution of the magnetohydrodynamic equations,” *J. Comput. Phys.*, vol. 35, no. 3, pp. 426-430, May 1980, doi: 10.1016/0021-9991(80)90079-0.

[19] H. K. Versteeg, W. Malalasekera, G. Orsi, J. H. Ferziger, A. W. Date, and J. D. Anderson, *An Introduction to Computational Fluid Dynamics - The Finite Volume Method*. 1995.

[20] A. Tassone, “CFD with OpenSource software Magnetic induction and electric potential solvers for incompressible MHD flows.” [Online]. Available: http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2016.

[21] C. Greenshields, “OpenFOAM Programmer’s Guide,” 2011.

[22] ANSYS inc., *ANSYS Fluent User’s Guide*, 15th ed. 2013.

[23] G. Witvoet, M. Steinbuch, E. Westerhof, and N. Doelman, “Modeling and simulating the sawtooth instability in nuclear fusion.,” Jan. 2009, Conference: Benelux Meeting on Systems and Control.

[24] M. Roberts, M. Leroy, J. Morales, W. Bos, and K. Schneider, “Self-organization of helically forced MHD flow in confined cylindrical geometries,” *Fluid Dyn. Res.*, vol. 46, no. 6, p. 061422, Dec. 2014, doi: 10.1088/0169-5983/46/6/061422.