國立成功大學
太空與電漿科學研究所
碩士論文

National Cheng Kung University Institute
of Space and Plasma Sciences Master Thesis

考量真空腔渦電流影響下之mini-Tokamak中心螺線管電流之設計

Design of the Central Solenoid Current Profile for the mini-Tokamak Including the Effects of Eddy Currents in the Vacuum-Vessel Wall

研究生 (Author)： 畢永葳 Yung-Wei Pi
指導老師 (Advisor)： 張博宇博士 Dr. Po-Yu Chang
中華民國一百一十四年六月 June, 2025

國立成功大學

National Cheng Kung University

碩士論文合格證明書

Certificate of Approval for Master's Thesis

題目： 考量真空腔渦電流影響下之mini-Tokamak中心螺線管電流之設計

Title： Design of the Central Solenoid Current Profile for the mini-Tokamak Including the Effects of Eddy Currents in the Vacuum-Vessel Wall

研究生：畢永葳

本論文業經審查及口試合格特此證明

This is to certify that the Master's Thesis of YUNG-WEI PI

論文考試委員：

has passed the oral defense under the decision of the following committee members

曲克宇　張博宇　河森榮一郎

指導教授 Advisor(s)： 張博宇

單位主管 Chair/Director： 電漿所所長張博宇

（單位主管是否簽章授權由各院、系（所、學位學程）自訂 ）

(The certificate must be signed by the committee members and advisor(s). Each department/graduate institute/degree program can determine whether the chair/director also needs to sign.)

2025/6/24

摘要

本論文旨在設計 mini-Tokamak 的中心螺線管電流波形，以產生電漿擊穿所需的環形迴路電壓，並進一步驅動電漿電流至 10 kA，同時將電漿加熱至 10 eV。隨時間變化的中心螺線管電流會在真空腔的內外壁中誘發渦電流，而這些渦電流將影響迴路電壓。為此，我們建立了一套數值模型，可用來計算托克馬克各組件的電阻與電感，並求得真空腔壁中感應的渦電流及其對迴路電壓的影響。模擬結果顯示，渦電流在初期會抑制迴路電壓，但其效應將迅速衰減，在 0.5 ms 後影響降至 5% 以下。因此，中心螺線管電流的作用時間應超過 0.5 ms，以降低渦電流干擾。此外，我們建立了一個模型，用以計算電漿電流與溫度在歐姆加熱下的變化。電漿電阻採用新古典理論計算，而電離率則使用一套混合模型求得，此模型結合經驗公式與 ADAS 資料庫中的係數，可描述從弱電離狀態至完全電離過程中的電離行為。為在粒子密度 $n = 10^{17}\ m^{-3}$（對應氣壓約 10^{-5} Torr）且磁力線連接長度為 500 m 的條件下實現擊穿，所需的環形迴路電壓約為 0.046 V。在此條件下，中心螺線管的電流變化率需大於 289 A/ms。為使 mini-Tokamak（主半徑 85 mm、短半徑 55 mm、橢圓延展率 $\kappa = 1.82$）中的電漿溫度達到 10 eV，中心螺線管電流應先於 10 ms 內線性由 5.4 kA 降至 0 kA，再於接續 40 ms 內由 0 kA 緩降至 −1.5 kA。在不考慮粒子流失及其所造成的能量流失的條件下，此中心螺線管電流波形可使電漿溫度在 40 ms 內維持於 10 eV。

關鍵字：托克馬克、磁約束核融合、渦電流、電漿擊穿

Abstract

This thesis presents the design of the central solenoid current profile of the mini-Tokamak to generate the toroidal loop voltage required for plasma breakdown and to subsequently drive the plasma current to 10 kA and heat the plasma to 10 eV. The time-varying current induces eddy currents in the inner and outer vacuum-vessel walls, which in turn affect the loop voltage. A numerical model is developed to compute the resistance and inductance of the tokamak components, the induced eddy currents and the loop voltage, with the effects of eddy currents in the vacuum-vessel wall included. The calculations show that eddy currents initially suppress the loop voltage but decay rapidly, with their influence reducing to below 5% after 0.5 ms. Consequently, the solenoid current should be maintained longer than 0.5 ms to minimize eddy current effects. In addition, a time-dependent plasma model is constructed to calculate plasma current and temperature evolution due to Ohmic heating. Plasma resistance is calculated based on neoclassical resistivity, and the ionization fraction is obtained using a hybrid model that combines collisional model from empirical formula with coefficients from the ADAS database. This formulation enables estimation of ionization behavior from the initial weakly ionized state to a fully ionized plasma. To achieve breakdown at a particle density of $n = 10^{17}\ m^{-3}$ (corresponding to a pressure of 10^{-5} Torr) with a connection length of 500 m, a loop voltage of approximately 0.046 V is required. Under these conditions, the central solenoid current change rate must exceed 289 A/ms. Furthermore, to achieve a plasma temperature of 10 eV in the mini-Tokamak with a major radius of 85 mm, minor radius of 55 mm, and elongation $\kappa = 1.82$, the central solenoid current should decrease linearly from 5.4 kA to 0 kA in 10 ms, followed by a ramp from 0 kA to –1.5 kA in 40 ms. When the particle losses and the corresponding energy losses are neglected, the plasma temperature can be maintained at 10 eV for 40 ms.

Keywords: Tokamak, Magnetic confinement fusion, Eddy current, Plasma breakdown

誌謝

在大學至碩士這段期間，我最感謝博宇老師，我常常驚嘆於老師的教學熱誠，總是細心地指導每一位學生，並給予我們莫大的支持與鼓勵。我由衷地感謝老師給予我多次寶貴的學習機會，引領我對科學和核融合產生興趣。希望未來某天，我能夠成為像老師一樣厲害的人。我也非常感謝河森榮一郎老師的教導，雖然每次上課回答不出問題都讓老師很頭痛，但老師每次仍舊耐心地引導我們。老師讓我學習到不同的思考邏輯，引導我從另一個角度看待問題，我期許自己未來能夠學會像老師那樣的思考方式。此外，也要感謝系辦的 Dabby 和安成，總是協助我處理各種瑣事，並提醒我重要時程。

感謝德岳在我口試期間的協助，以及一路以來在各方面給予的幫助與支持；也感謝奕佐在我每次迷惘時，總是毫無保留地給出真誠而深刻的建議。和你們一起出遊的時光，讓我在大四與研究所這段時間，依然能像大一那樣地瘋玩。感謝我大學的好朋友奕翔，總是無條件地給予我支持，如果當初你沒有提議來電漿所做專題，我或許就不會有機會接觸到核融合與博宇老師。同時，我也非常感謝我的父母，在我求學的路上給予我極大的自由與信任，始終支持我的選擇，讓我能夠無後顧之憂地成長。

Contents

List of Tables

List of Figures

1. Introduction

Nuclear fusion is a process in which two or more atomic nuclei fuse together forming a heavier nucleus. The mass reduction of the products compared to the reactants is released as energy. One example is the process in the Sun, as shown in Figure 1(a). It illustrates the proton–proton chain reaction, in which hydrogen nuclei fuse through a series of steps to form helium and release energy. This is the dominant reaction that powers with stars the size of the Sun or smaller. Another example is the reaction between a deuterium and a tritium, which generates a helium and a neutron, as shown in Figure 1(b):

$$ {}_{1}^{2}\mathrm{D} + {}_{1}^{3}\mathrm{T} \rightarrow {}_{2}^{4}\mathrm{He} + {}_{0}^{1}\mathrm{n}\ (+17.6\ \mathrm{MeV})\ . \qquad (1) $$

This reaction is considered the most favorable fusion reaction due to the highest cross-section among all function reactions at achievable temperatures.

This chapter consists of three sections. Brief introduction of nuclear fusion and the goal of this thesis will be given. Section 1.1 will introduce the two leading methods to achieve

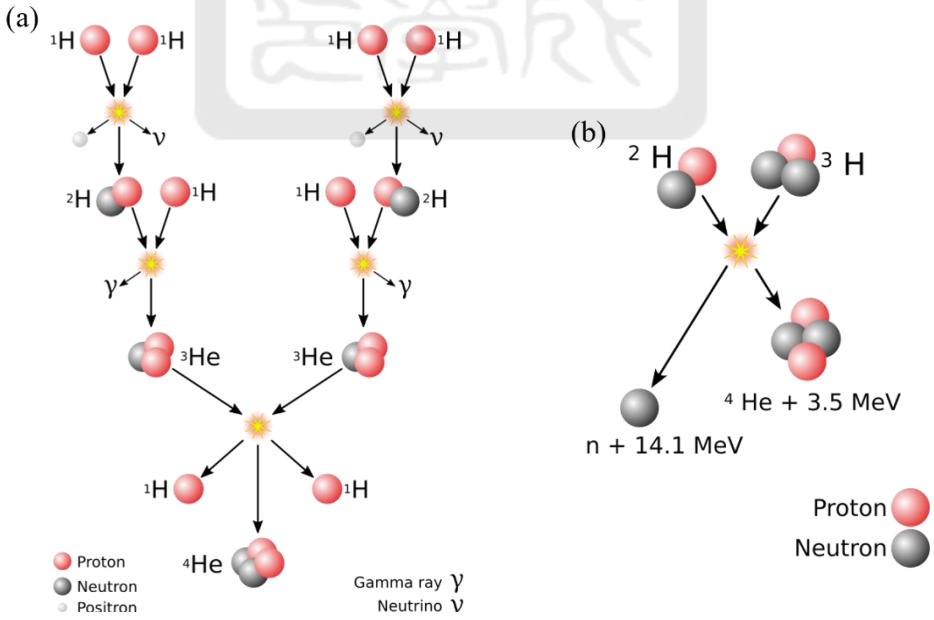

Figure 1: (a) Proton–proton chain reaction in stellar fusion[1]. (b) Deuterium–tritium (D–T) fusion reaction[2].

nuclear fusion in date. Next, Section 1.2 will explain the motivation of this thesis. Finally, Section 1.3 will present the goal of this thesis.

1.1. Nuclear fusion

To achieve fusion on Earth, it is necessary to create the extreme temperature, pressure, and confinement conditions required for fusion to occur. Two leading approaches are being developed: Inertial Confinement Fusion (ICF) and Magnetic Confinement Fusion (MCF). Each method relies on a different mechanism to confine and heat the fusion fuel to the required conditions.

1.1.1. Inertial confinement fusion (ICF)

The principle of inertial confinement fusion is shown in Figure 2. When laser beams are focused onto the surface of a spherical fuel capsule, the outer layer is rapidly heated and ablated, forming a plasma envelope, as shown by the red-yellow outer layer in Figure 2 (a). The blowoff of this hot plasma creates a reaction force that compresses the inner fuel symmetrically. As compression proceeds, the core temperature and density rise rapidly, eventually reaching the conditions required for fusion ignition. In the ignition phase, the fuel core reaches approximately 20 times its original density and a temperature of around 10^8 ℃.

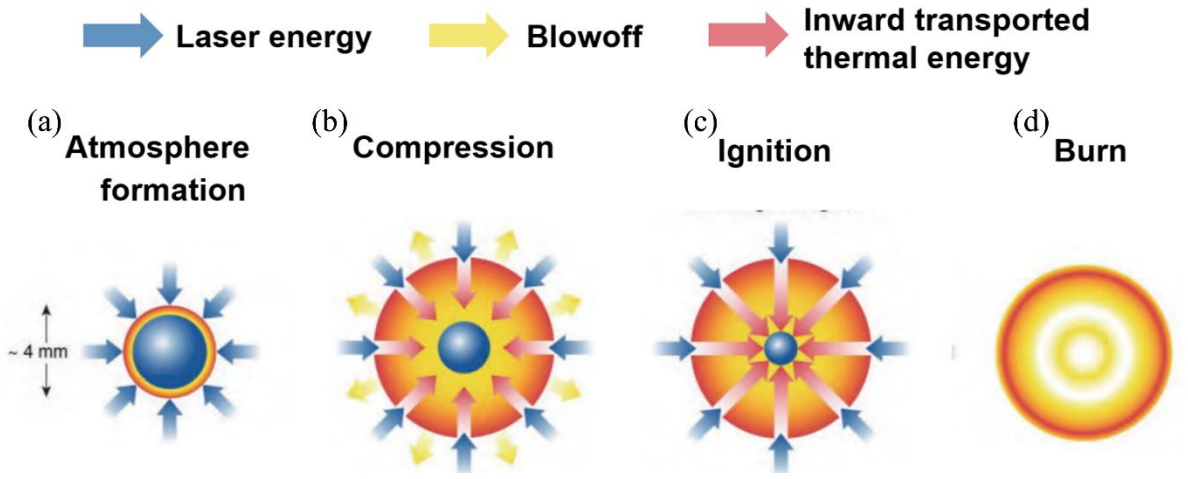

Figure 2: Stages of the Inertial Confinement Fusion Process[3].

Once fusion reactions begin at the center of the compressed target, thermal energy is transported outward, sustaining a burn wave that spreads outward and releases energy several times greater than the initial laser input.

1.1.2. Magnetic confinement fusion (MCF)

The configuration of magnetic confinement fusion is shown in Figure 3(a). In a typical MCF device, such as a tokamak, the plasma is confined in a toroidal (doughnut-shaped) configuration. These devices typically have an aspect ratio (R/a) larger than 2, where R is the major radius and a is the minor radius of the plasma, as shown as the outer plasma in Figure 3(b). In contrast, the spherical tokamak, shown as the inner plasma in Figure 3(b), features an apple-like shape with a much smaller aspect ratio smaller than 2.

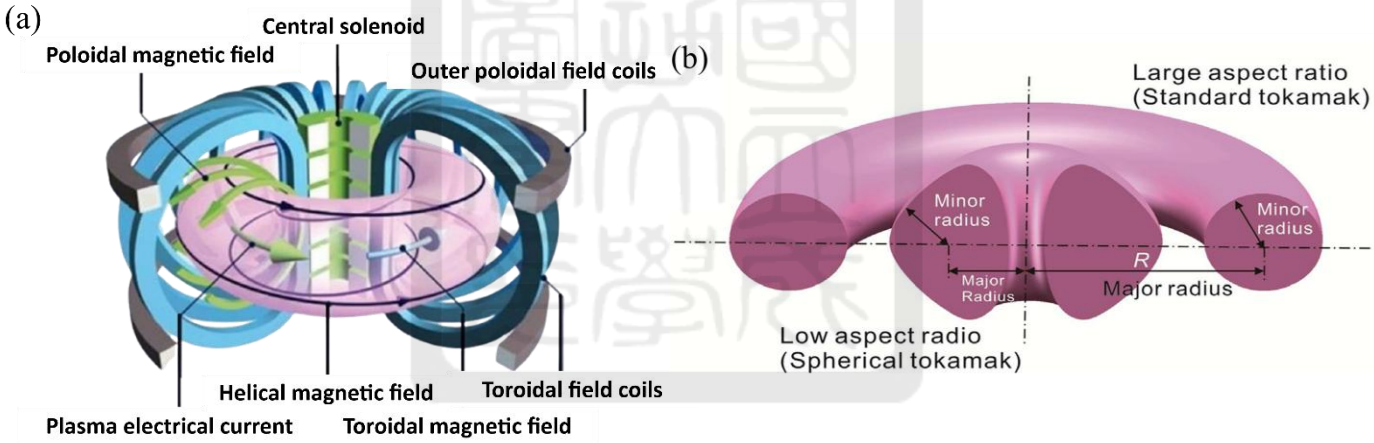

Figure 3: (a) Magnetic field structure in a tokamak[4]. (b) Comparison of spherical tokamak to standard tokamak[5].

Magnetic confinement in these devices is achieved through the combination of a toroidal magnetic field, generated by toroidal field coils (TFC), and a poloidal magnetic field, which is produced by a plasma current in the toroidal direction. This plasma current is typically induced by a central solenoid. Together, these fields create helical magnetic field lines, illustrated by the black spiral in Figure 3(a), that effectively traps plasma particles and

guide their motion within the confined region. At the same time, plasma is heated by varies sources, e.g., Ohmic heating, RF heating, neutral beam injection, etc. This configuration allows the plasma to be maintained at high temperature for a sufficiently long period to enable fusion reactions, in which a large amount of energy is released, such as in the D–T reaction shown in Figure 1(b). The mini-Tokamak developed in this thesis adopts the principle of magnetic confinement fusion, and its plasma takes the form of a spherical tokamak. An image of the mini-Tokamak is shown in Figure 4(a), and detailed information is provided in Chapter 2. Likewise, the Formosa Integrated Research Spherical Tokamak (FIRST), as shown in Figure 4(b) and introduced in Appendix A.15, is also a spherical tokamak, but on a larger scale. It is being developed under the support of National Science and Technology Council and for more advanced experiments.

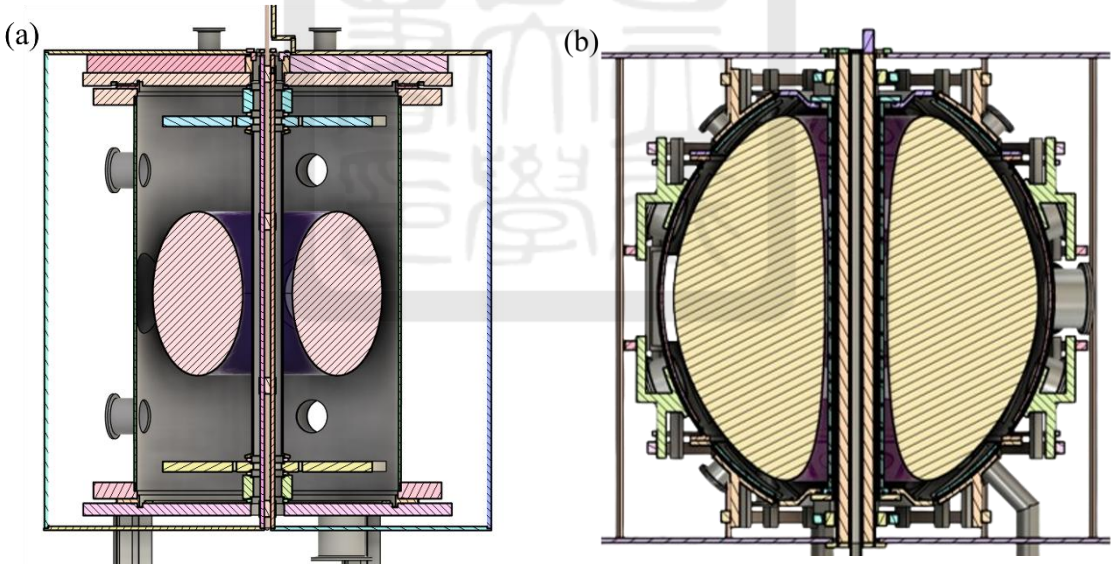

Figure 4: (a) Cross-sectional CAD drawings of the mini-Tokamak[6]. (b) Cross-sectional CAD drawings of the FIRST[6].

1.2. Motivation for developing the mini-Tokamak

The mini-Tokamak developed in this thesis serves as a versatile and accessible experimental platform for both educational applications and technical validation. Its primary purpose is to support experimental courses related to plasma physics and nuclear fusion, providing students with opportunities not only to learn theoretical concepts, but also to engage in hands-on experience. Through real-time observation, data acquisition, and analysis of plasma experiments, students can deepen their understanding of fusion phenomena and enhance their practical skills in diagnostics and experimental techniques.

In addition to its educational value, the mini-Tokamak also plays a key technical role in supporting the development of the Formosa Integrated Research Spherical Tokamak (FIRST), the first tokamak being designed and constructed in Taiwan. FIRST is expected to achieve first plasma by 2026, with the vacuum vessel scheduled for installation in early 2026. However, several key components, such as the power supplies and the current drivers for the central solenoid (CS), poloidal field coils (PFCs), and toroidal field coils (TFCs) are expected to arrive earlier. In this context, the mini-Tokamak can be used as a preliminary testing platform for FIRST. It allows for early validation of current drivers and magnetic field measurement systems under tokamak-relevant conditions. By conducting these preparatory tests in advance, key systems can be properly configured and made ready for operation. This enables a smooth transition to full-scale testing once FIRST is installed, maximizing efficiency and minimizing commissioning delays.

1.3. The goal of this thesis

The goal of this thesis is to design the central solenoid (CS) current profile for the mini-Tokamak. To achieve plasma breakdown and drive the plasma current, the CS current must be carefully tailored to provide the required loop voltage under constrained conditions. The time-varying current in the central solenoid not only induces voltage in the vacuum vessel

to initiate plasma breakdown but also generates eddy currents in the surrounding vacuum-vessel walls. These eddy currents can significantly alter the distribution of loop voltage and affect the efficiency of plasma initiation. Furthermore, once the plasma is formed, the evolution of plasma parameters, such as plasma current and temperature, will further influence the required CS current profile.

To account for these effects, this thesis develops a sequence of models to calculate and analyze the various factors influencing the design of the CS current profile. A detailed description of the mini-Tokamak specifications and geometry will first be provided in Chapter 2. Next, Chapter 3 introduces the calculation of eddy currents in the vacuum-vessel walls, based on the chamber's geometry and material properties. Chapter 4 presents the computation of the loop voltage induced by the time-varying CS current, with the influence of eddy currents included. Chapter 5 focuses on the calculation of the evolution of plasma parameters, such as temperature, resistance, and plasma current, under the designed CS current profile. Future work and conclusion are presented in Chapters 6 and 7, respectively. In addition to the mini-Tokamak application, the modeling framework developed in this thesis is further applied to the preliminary current profile design for the Formosa Integrated Research Spherical Tokamak (FIRST), with detailed calculations provided in Appendix A.15.

2. mini-Tokamak specification

Figure 5(a) shows a simplified cross-sectional view of the mini-Tokamak in the xz plane, while Figure 5(b) presents a schematic of the right half of the vacuum vessel structure, referenced from the device's centerline. As illustrated in Figure 5(a) and (b), the vacuum vessel is a cylinder, with a rectangular cross-section in the xz plane. The inner wall of the vacuum vessel is made of a standard KF 40 nipple with 1.8 mm in thickness. The inner edge is located at 17.25 mm from the centerline, and the vertical height is 531 mm. The outer wall is 3 mm thick, with its inner edge locates at 160 mm from the centerline, and the vertical height is 531 mm. The top and bottom covers, with a thickness of 16 mm and 15 mm, respectively, connect the inner and outer walls and are treated as part of the outer wall in eddy current calculations. Therefore, the height of both the inner and outer vacuum-vessel walls is considered to be 531 mm in the following calculations.

Regarding the plasma geometry, as shown in Figure 5(a), the plasma has a major radius of 85 mm and a minor radius of 55 mm. The elongation is set to 1.82, resulting in a plasma cross-section with a long axis of approximately 100.1 mm and a short axis of 55 mm. To ensure confinement, the gyro radius of the particles in the mini-Tokamak must be smaller

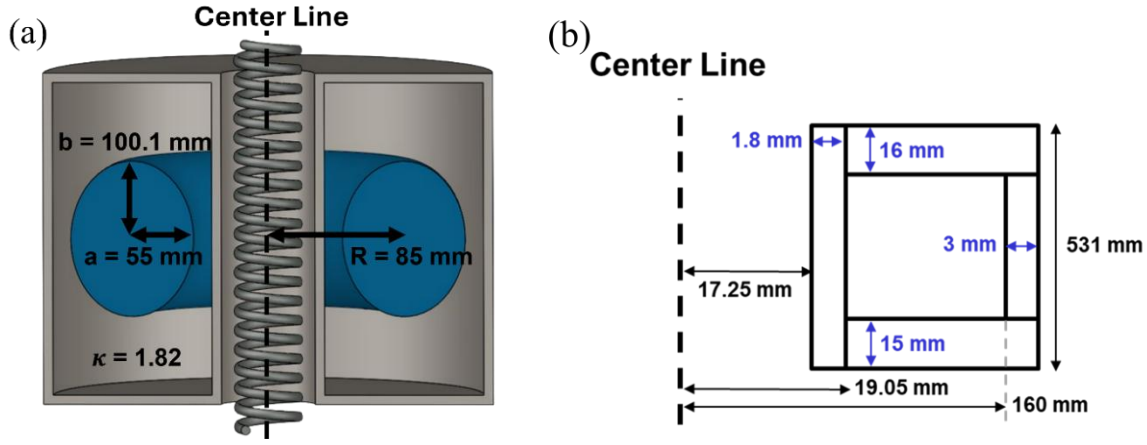

Figure 5: (a) Simplified xz-plane cross-section of mini-Tokamak. (b) Structural schematic of the vacuum-vessel of the mini-Tokamak.

than the distances between the plasma edge and the vacuum vessel walls, which are 11 mm on the inner side and 17 mm on the outer side. Otherwise, the particles will collide with the wall and be lost. Helium is selected as the working gas, and the plasma temperature is limited to 10 eV, with detailed calculations provided in Appendix A.1.

There will be four rectangular toroidal field coils with a current of 4 kA, a toroidal magnetic field of 0.1 T at 85 mm can be produced. It is intended to achieve a plasma current of up to 10 kA and a plasma temperature of approximately 10 eV. The mini-Tokamak has six pairs, totaling twelve rectangular poloidal field coils. The position and current of each coil are given in Appendix A.2.

The mini-Tokamak employs a central solenoid (CS), located inside the inner-vacuum-vessel wall, as shown in Figure 5(a) and illustrated in detail in Figure 6. The CS consists of two layers, each comprising 92 turns of copper conductor, giving a total of 184 turns. The conductor is a solid copper wire with a circular cross-section of 2.6 mm in diameter, and an insulation layer of 1.7 mm in thickness, resulting in a total outer diameter of 6 mm per turn. The CS will wrap around the inner legs of the toroidal field coils and be fitted within the inner-vacuum-vessel wall.

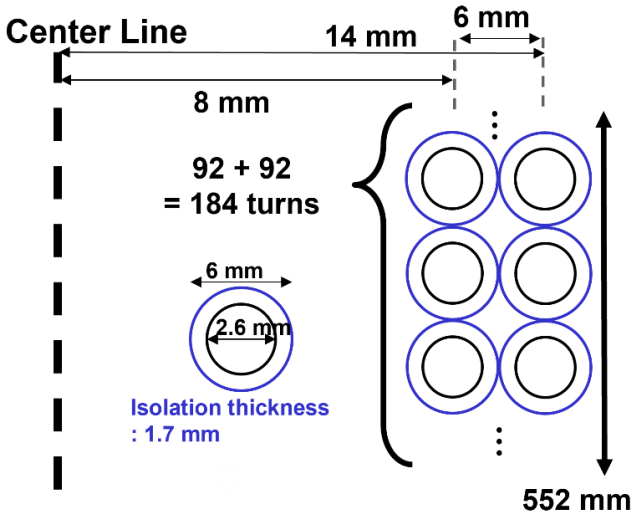

Figure 6: Specifications of the central solenoid in the mini-Tokamak.

3. Calculation of the eddy currents in the inner and outer vacuum-vessel walls

In the tokamak device, we generate plasma, drive plasma current, and control plasma shape by adjusting the time-varying currents in coils such as poloidal field coil (PFC), toroidal field coil (TFC), and central solenoid (CS). However, the time-varying magnetic fields generated by these coils will induce eddy currents in vacuum-vessel walls, which can impact the originally calculated magnetic fields. Therefore, before constructing the tokamak device, it is necessary to include the eddy currents within the device in the design to obtain more accurate results.

This chapter consists of six sections. Section 3.1 introduces the full circuit equation used to calculate the induced currents between the central solenoid and vacuum-vessel walls. Section 3.2 explains the conjugate gradient (CG) method employed to obtain a numerical stable and converge solution. Section 3.3 describes the components defined in the circuit model. Section 3.4 presents the calculation of inductance and resistance for each components. Section 3.5 shows the calculated results, and Section 3.6 provides the conclusion.

3.1. The full circuit equation

To calculate the eddy currents in the vacuum-vessel walls, we can treat the vacuum-vessel wall as a “coil”. Then, the eddy currents in the vacuum-vessel walls can be calculated using the mutual inductance between the vacuum-vessel walls and coils. In particular, the central solenoid is considered here. To calculate the eddy currents, we consider the vacuum-vessel wall as many ring coils stacking on top of each other. Then, by calculating the induced current in each ring coil, called the vacuum-vessel element, we can calculate the eddy currents at different vacuum-vessel elements using the full circuit equation[7]:

$$\vec{V} = \mathbf{M}\frac{d\vec{I}}{dt} + \mathbf{R}\vec{I}\,. \tag{2}$$

Here, $\vec{V}$ represents the voltage across each element/coil, $\vec{I}$ represents the current flowing through each element/coil, $\mathbf{M}$ represents inductance of each element/coil, and $\mathbf{R}$ represents resistance of each element/coil. This equation states that the total voltage "$\vec{V}$" in a closed loop equals its induced voltage "$\mathbf{M}\frac{d\vec{I}}{dt}$" plus the voltage drop due to resistance "$\mathbf{R}\vec{I}$". In Chapter 5, plasma is treated as a ring coil and can be included in the calculation.

For convenience in calculations, we have $\vec{V}$ and $\vec{I}$ in the vector form to represent the voltage and the current in each coil and each vacuum-vessel element. On the other hand, $\mathbf{M}$ and $\mathbf{R}$ are written in the matrix form including the self-inductance and the resistance of each coil and each vacuum-vessel element, and the mutual inductance between each component. In addition, we use subscribes c, v, p to represent different types of components. Subscribe c represents coils in the tokamak device, such as the central solenoid, PFC, and TFC; subscribe v represents elements in the tokamak device's vacuum vessel; subscribe p represents the plasma. Therefore, $\vec{I}$ is a column vector of size $n+1$, representing the currents of the total n components of coils and vacuum-vessel elements in the tokamak device plus the plasma current; $\vec{V}$ is also a column vector of size $n+1$, representing the voltages of the total n components of coils and vacuum-vessel elements in the tokamak device plus the loop voltage of plasma; $\mathbf{R}$ is a diagonal matrix of size $(n+1)\times(n+1)$, representing the resistance of the total n components of coils and vacuum-vessel elements plus the plasma; $\mathbf{M}$ is a symmetric matrix of size $(n+1)\times(n+1)$, representing the self-inductance (diagonal) of the total n components of coils and vacuum-vessel elements plus plasma, and the mutual inductance (off-diagonal) between them. Therefore, Eq. (2) can be written explicitly as:

$$\vec{V} = \mathbf{M}\frac{d\vec{I}}{dt} + \mathbf{R}\vec{I}$$

$$\Rightarrow \begin{bmatrix} \vec{V}_{\mathrm{c1}} \\ \vec{V}_{\mathrm{c2}} \\ \vdots \\ \vec{V}_{\mathrm{v1}} \\ \vec{V}_{\mathrm{v2}} \\ \vdots \\ \vec{V}_{\mathrm{p}} \end{bmatrix} = \begin{bmatrix} L_{\mathrm{c1}} & M_{\mathrm{c1,c2}} & M_{\mathrm{c1,c3}} & \cdots & M_{\mathrm{c1,v1}} & M_{\mathrm{c1,v2}} & M_{\mathrm{c1,v3}} & \cdots & M_{\mathrm{c1,p}} \\ M_{\mathrm{c2,c1}} & L_{\mathrm{c2}} & M_{\mathrm{c2,c3}} & \cdots & M_{\mathrm{c2,v1}} & M_{\mathrm{c2,v2}} & M_{\mathrm{c2,v3}} & \cdots & M_{\mathrm{c2,p}} \\ M_{\mathrm{c3,c1}} & M_{\mathrm{c3,c2}} & L_{\mathrm{c3}} & \cdots & M_{\mathrm{c3,v1}} & M_{\mathrm{c3,v2}} & M_{\mathrm{c3,v3}} & \cdots & M_{\mathrm{c3,p}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ M_{\mathrm{v1,c1}} & M_{\mathrm{v1,c2}} & M_{\mathrm{v1,c3}} & \cdots & L_{\mathrm{v1}} & M_{\mathrm{v1,v2}} & M_{\mathrm{v1,v3}} & \cdots & M_{\mathrm{v1,p}} \\ M_{\mathrm{v2,c1}} & M_{\mathrm{v2,c2}} & M_{\mathrm{v2,c3}} & \cdots & M_{\mathrm{v2,v1}} & L_{\mathrm{v2}} & M_{\mathrm{v2,v3}} & \cdots & M_{\mathrm{v2,p}} \\ M_{\mathrm{v3,c1}} & M_{\mathrm{v3,c2}} & M_{\mathrm{v3,c3}} & \cdots & M_{\mathrm{v3,v1}} & M_{\mathrm{v3,v2}} & L_{\mathrm{v3}} & \cdots & M_{\mathrm{v3,p}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ M_{\mathrm{p,c1}} & M_{\mathrm{p,c2}} & M_{\mathrm{p,c3}} & \cdots & M_{\mathrm{p,v1}} & M_{\mathrm{p,v2}} & M_{\mathrm{p,v3}} & \cdots & L_{\mathrm{p}} \end{bmatrix} \begin{bmatrix} \frac{d\vec{I}_{\mathrm{c1}}}{dt} \\ \frac{d\vec{I}_{\mathrm{c2}}}{dt} \\ \vdots \\ \frac{d\vec{I}_{\mathrm{v1}}}{dt} \\ \frac{d\vec{I}_{\mathrm{v2}}}{dt} \\ \vdots \\ \frac{d\vec{I}_{\mathrm{p}}}{dt} \end{bmatrix}$$

$$+ \begin{bmatrix} R_{\mathrm{c1}} & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & R_{\mathrm{c2}} & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & R_{\mathrm{v1}} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & R_{\mathrm{v2}} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & R_{\mathrm{p}} \end{bmatrix} \begin{bmatrix} \vec{I}_{\mathrm{c1}} \\ \vec{I}_{\mathrm{c2}} \\ \vdots \\ \vec{I}_{\mathrm{v1}} \\ \vec{I}_{\mathrm{v2}} \\ \vdots \\ \vec{I}_{\mathrm{p}} \end{bmatrix}. \quad (3)$$

Notice that no external voltages are applied on the vacuum-vessel elements and the plasma: $\vec{V}_{\mathrm{v1}} = \vec{V}_{\mathrm{v2}} = \cdots = \vec{V}_{\mathrm{p}} = 0$. The only non-zero voltage are $\vec{V}_{\mathrm{c1}}$, $\vec{V}_{\mathrm{c2}}$, ….

Eq. (2) can be solved numerically using a finite difference method:

$$\vec{V} = \mathbf{M}\frac{d\vec{I}}{dt} + \mathbf{R}\vec{I} \Rightarrow \vec{V} = \mathbf{M}\frac{\vec{I'} - \vec{I}}{\Delta t} + \mathbf{R}\vec{I}$$

$$\Rightarrow \vec{V} = \mathbf{M}\frac{\vec{I'}}{\Delta t} - \mathbf{M}\frac{\vec{I}}{\Delta t} + \mathbf{R}\vec{I} = \mathbf{M}\frac{\vec{I'}}{\Delta t} + \left(\mathbf{R} - \frac{\mathbf{M}}{\Delta t}\right)\vec{I} \Rightarrow \frac{\mathbf{M}}{\Delta t}\vec{I'} = \vec{V} + \left(\frac{\mathbf{M}}{\Delta t} - \mathbf{R}\right)\vec{I}$$

$$\Rightarrow \vec{I'} = \left(\frac{\mathbf{M}}{\Delta t}\right)^{-1}\left[\vec{V} + \left(\frac{\mathbf{M}}{\Delta t} - \mathbf{R}\right)\vec{I}\right]. \quad (4)$$

In Eq. (4), $\vec{I'}$ represents the current in the following time step of $\vec{I}$. Notice that matrix $\mathbf{M}$ and $\mathbf{R}$ are constant when plasma is not included because coils and vacuum-vessel elements are fixed once the tokamak is built. By providing $\vec{V}(t)$, we can obtain $\vec{I}(t)$ at any time.

When using the full circuit equation to calculate the eddy currents induced by the central solenoid in the vacuum-vessel walls, the feedback effect of the eddy currents on the induced current in the central solenoid is negligible. This is because the eddy currents are relatively small compared to the central solenoid's driving current. To simplify the calculation and enhance computational efficiency, we disregard this feedback effect in our calculation as shown in Eq. (5). Our current calculations only include the central solenoid (subscript cs), the vacuum-vessel walls (subscript v), and the plasma (subscript p). Therefore, Eq. (5) only accounts for these components.

$$-\begin{bmatrix} M_{\mathrm{v1,cs}} \\ M_{\mathrm{v2,cs}} \\ M_{\mathrm{v3,cs}} \\ \vdots \\ M_{\mathrm{p,cs}} \end{bmatrix} \frac{d\vec{I}_{cs}}{dt} = \begin{bmatrix} L_{\mathrm{v1}} & M_{\mathrm{v1,v2}} & M_{\mathrm{v1,v3}} & \cdots & M_{\mathrm{v1,p}} \\ M_{\mathrm{v2,v1}} & L_{\mathrm{v2}} & M_{\mathrm{v2,v3}} & \cdots & M_{\mathrm{v2,p}} \\ M_{\mathrm{v3,v1}} & M_{\mathrm{v3,v2}} & L_{\mathrm{v3}} & \cdots & M_{\mathrm{v3,p}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ M_{\mathrm{p,v1}} & M_{\mathrm{p,v2}} & M_{\mathrm{p,v3}} & \cdots & L_{\mathrm{p}} \end{bmatrix} \begin{bmatrix} \vec{I}'_{\mathrm{v1}} \\ \vec{I}'_{\mathrm{v2}} \\ \vdots \\ \vec{I}'_{\mathrm{p}} \end{bmatrix} + \begin{bmatrix} R_{\mathrm{v1}} & 0 & \cdots & 0 \\ 0 & R_{\mathrm{v2}} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & R_{\mathrm{p}} \end{bmatrix} \begin{bmatrix} \vec{I}_{\mathrm{v1}} \\ \vec{I}_{\mathrm{v2}} \\ \vdots \\ \vec{I}_{\mathrm{p}} \end{bmatrix}. \tag{5}$$

As mentioned earlier, $\vec{V}$ represents the external voltage. Since the vacuum-vessel walls and plasma are passive elements, which don't have an independent external voltage source, their voltage is set to $V = 0$. However, since the feedback effect on the central solenoid is ignored, we can treat induced effect on the other components as their respective source term. This induced effect is expressed as the mutual inductance between each component and central solenoid times the rate of change of the central solenoid's current. For example, as shown in Eq. (5), $-M_{\mathrm{v1,cs}} \cdot \frac{dI_{cs}}{dt}$ represents the induced voltage from the central solenoid on the first vacuum-vessel wall element.

To solve Eq. (5) numerically, conjugate gradient (CG) method is used, which will be introduced in next section.

3.2. Conjugate gradient (CG) method

Currently, our calculation involves a matrix composed of 491 components, which will be described in detail in the next section. This matrix will continue to grow as more components are included in the model. This increase in matrix size will lead to greater computational complexity and longer computation times. To efficiently handle these challenges, instead of using conventional iterative methods, we employ the Conjugate Gradient (CG) method[8].

Eq. (2) can be written as a simple linear system:

$$\mathbf{A}\vec{x} = \vec{b} \tag{6}$$

where $\mathbf{A}$ is a symmetric matrix.

To solve $\vec{x}$, basic iterative methods proceed as follows:

(1) Initial guess: Start with an initial guess $\vec{x}_0$.

(2) Residual calculation: Compute the residual $\vec{r}_0 = \vec{b} - \mathbf{A}\vec{x}_0$, which measures the difference between the current approximation and the true solution which equals to $\vec{b}$.

(3) Update the solution: Use the residual to update the approximation. For example, in the Richardson method: $\vec{x}_{i+1} = \vec{x}_i + \omega\vec{r}_i$, where ω is a relaxation parameter chosen to ensure convergence.

(4) Convergence check: Evaluate the norm of the residual $||\vec{r}_i||$. If it is below a specified tolerance, the solution is accepted. Otherwise, repeat steps (2)-(4).

In contrast, the CG method, described subsequently, improves upon basic iterative methods by using not only residuals but also additional conjugate directions to ensure faster and more reliable convergence.

To apply the CG method, we first reformulate the original linear system $\mathbf{A}\vec{x} = \vec{b}$ as an optimization problem. Specifically, we introduce the quadratic functional:

$$f(\vec{x}) = \frac{1}{2}\vec{x}^{\mathrm{T}}\mathbf{A}\vec{x} - \vec{b}^{\mathrm{T}}\vec{x} + C\,. \tag{7}$$

To better understand this formulation, consider a specific example[8]:

$$\vec{x} = \begin{bmatrix}2\\1\end{bmatrix}, \mathbf{A} = \begin{bmatrix}2 & 0\\0 & 4\end{bmatrix}, \vec{b} = \begin{bmatrix}1\\-1\end{bmatrix}, C = 0\,. \tag{8}$$

Substituting these into the function $f(\vec{x})$:

$$f = \frac{1}{2}\begin{bmatrix}2 & 1\end{bmatrix}\begin{bmatrix}2 & 0\\0 & 4\end{bmatrix}\begin{bmatrix}2\\1\end{bmatrix} - \begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}2\\1\end{bmatrix} + 0 = 5\,. \tag{9}$$

This shows that the function $f(\vec{x})$ takes a vector input and returns a scalar output.

When $\vec{x} \in \mathbb{R}^2$, the function $f(\vec{x})$ maps each point in the two-dimensional plane to a single real-valued output, which can be interpreted as height. As a result, the quadratic function defines a smooth three-dimensional surface, where the shape of the surface reflects how $f(\vec{x})$ varies with respect to $\vec{x}$. Figure 7 provides both 2D and 3D visualization of the function $f(\vec{x})$. Specifically, Figure 7(a) shows the contour lines (or level curves) of constant function values, illustrating the landscape from a top-down perspective, while Figure 7(b) presents the corresponding 3D surface, where the height at each point represents the value of $f(\vec{x})$.

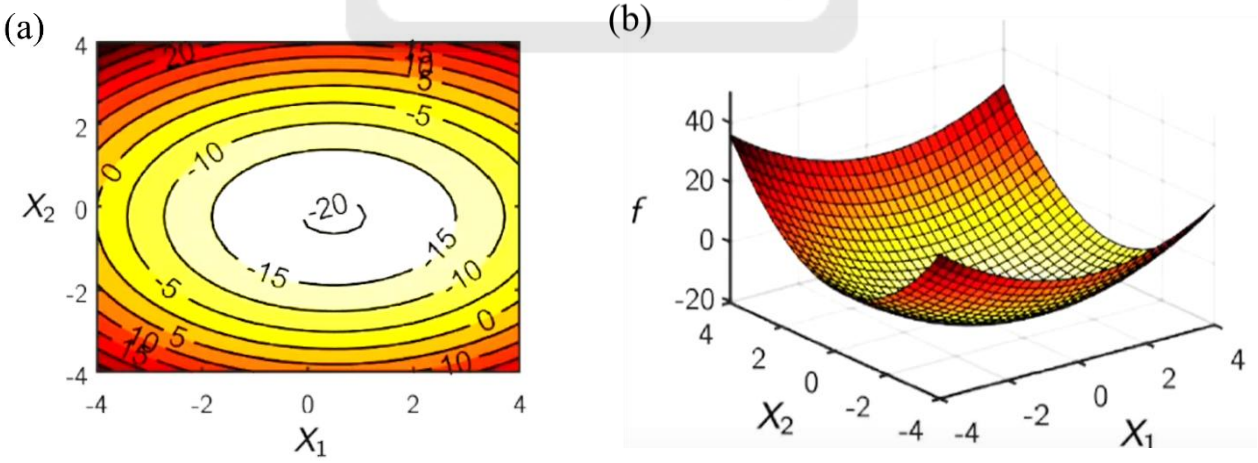

Figure 7: (a) Contour plot of quadratic function[8]. (b) 3D surface plot of quadratic function[8].

If we want to find the lowest point on this surface, the minimum occurs when the gradient of the function equals to zero:

$$\nabla f(\vec{x}) = \mathbf{A}\vec{x} = \vec{b} = 0\,. \tag{10}$$

Therefore, solving Eq. (10) is equivalent to finding the minimum of the quadratic functional given in Eq. (6).

Recall that the residual is defined as $\vec{r} = \vec{b} - \mathbf{A}\vec{x}$, which corresponds to the negative gradient of the objective function, $-\nabla f(\vec{x})$, evaluated at $\vec{x}$. In this sense, the residual provides a natural direction for descent in the solution space. Moreover, the constant term C does not affect the location of the minimum, since we are minimizing the gradient $\nabla f(\vec{x})$. Therefore, C can be ignored without loss of generality.

The CG method is to iteratively approach the minimum of the quadratic function $f\vec{x}$, which corresponds to the true solution $\vec{x}^*$ of the linear system $\mathbf{A}\vec{x} = \vec{b}$.

To do this, CG generates a sequence of iterates $\vec{x}_0, \vec{x}_1, \vec{x}_2, \ldots$, where each new point is computed by moving from the current point $\vec{x}_\mathrm{k}$ in a specific direction $\vec{p}_\mathrm{k}$, scaled by a step size α_k:

$$\vec{x}_{\mathrm{k}+1} = \vec{x}_\mathrm{k} + \alpha_\mathrm{k}\vec{p}_\mathrm{k}\,. \tag{11}$$

This update rule is visualized in Figure 8. The point $\vec{x}_\mathrm{k}$ is the current position, and $\vec{p}_\mathrm{k}$ is the direction in which we search. The scalar α_k controls how far we move along in that direction. The red line shows the direction $\vec{p}_\mathrm{k}$, and the product $\alpha_\mathrm{k}\vec{p}_\mathrm{k}$ gives the step vector that takes us from $\vec{x}_\mathrm{k}$ to the next point $\vec{x}_{\mathrm{k}+1}$.

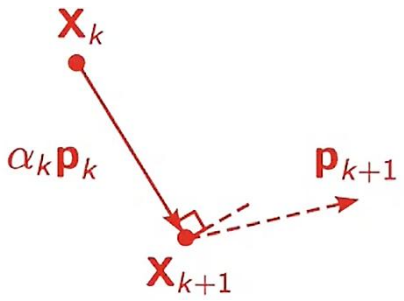

Figure 8: Visualization of a single CG iteration step[9].

In the CG method, we are interested in the iteration error, defined as:

$$\vec{e}_{\mathrm{k}} = \vec{x}_{\mathrm{k}} - \vec{x}^{*}\,, \tag{12}$$

where $\vec{x}^{*}$ is the true solution, $\vec{x}_{\mathrm{k}}$ is the current approximation at iteration k. Geometrically, $\vec{e}_{\mathrm{k}}$ is the vector pointing from the current estimate $\vec{x}_{\mathrm{k}}$ toward the true solution $\vec{x}^{*}$ $(\vec{x}_{\mathrm{final}})$, as shown Figure 9(a). If this vector were known, we could jump directly to the solution in one step. However, since $\vec{x}^{*}$ is unknown, we must iteratively reduce the error instead. Once the solution is updated to $\vec{x}_{\mathrm{k+1}}$, the new error becomes:

$$\vec{e}_{\mathrm{k+1}} = \vec{x}_{\mathrm{k+1}} - \vec{x}^{*}\,, \tag{13}$$

as shown in Figure 9(b). Using Eq. (11) and substituting into the expression for $\vec{e}_{\mathrm{k+1}}$:

$$\vec{e}_{\mathrm{k+1}} = (\vec{x}_{\mathrm{k}} + \alpha_{\mathrm{k}}\vec{p}_{\mathrm{k}}) - \vec{x}^{*} = (\vec{x}_{\mathrm{k}} - \vec{x}^{*}) + \alpha_{\mathrm{k}}\vec{p}_{\mathrm{k}} = \vec{e}_{\mathrm{k}} + \alpha_{\mathrm{k}}\vec{p}_{\mathrm{k}}\,. \tag{14}$$

This relation shows that the error is updated at each step based on the step size α_{k} and the search direction $\vec{p}_{\mathrm{k}}$. Moving from $\vec{x}_{\mathrm{k}}$ to $\vec{x}_{\mathrm{k+1}}$ causes the error vector to shorten and rotate closer to the true solution direction. To accelerate convergence, we want each new search direction $\vec{p}_{\mathrm{k}}$ to avoid retracing the directions used in previous steps. This is achieved through conjugate directions.

Before we delve further into the CG method, we first need to clarify what the term "conjugate" means in this context. Two vectors $\vec{a}$ and $\vec{b}$ are said to be A-conjugate (or

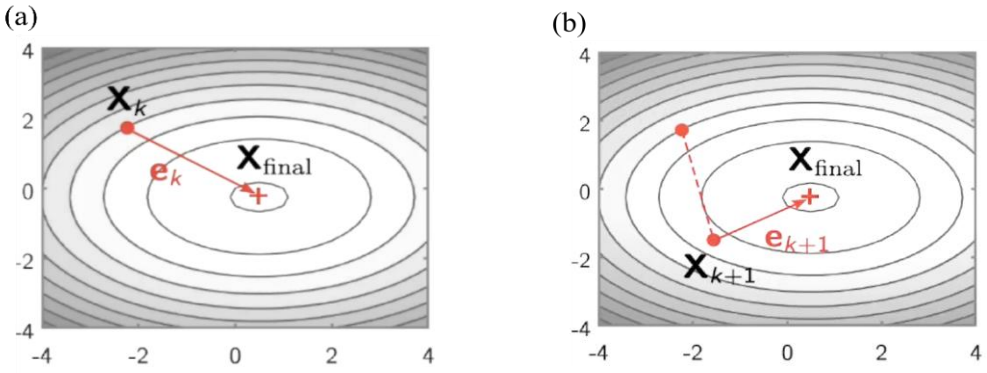

Figure 9: (a) Error vector at iteration k[8]. (b) Error vector at iteration $\mathrm{k}+1$[8].

conjugate with respect to the matrix $\mathbf{A}$), if the following condition holds for a symmetric positive definite matrix $\mathbf{A}$:

$$\vec{a}^{\mathrm{T}}\mathbf{A}\vec{b} = 0\,. \tag{15}$$

As shown in Figure 10(a), the vector $\mathbf{A}\vec{b}$ can be thought of as a transformation of $\vec{b}$ under the matrix $\mathbf{A}$. The conjugacy condition above implies that this transformed vector $\mathbf{A}\vec{b}$ is orthogonal to $\vec{a}$. This guarantees that each new direction explores a previously unvisited subspace of the solution space, avoiding inefficient zig-zagging patterns typical of steepest descent methods.

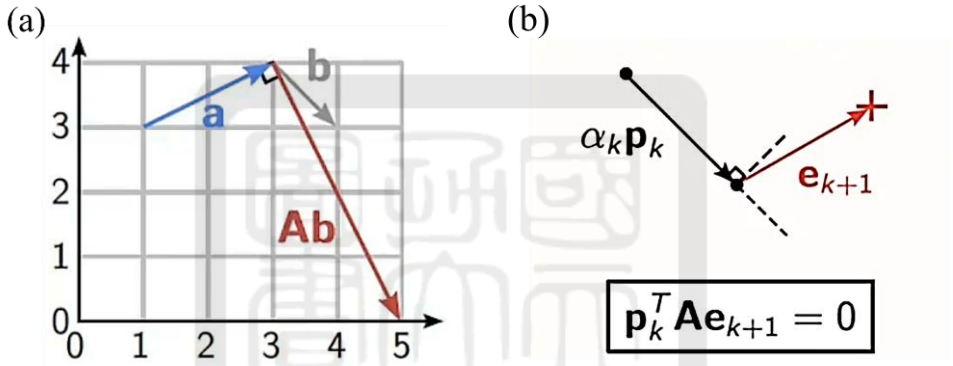

Figure 10: (a) Illustration of A-conjugacy between vectors[8]. (b) Stopping condition based on A-orthogonality[8].

Our goal is to determine suitable expressions for the scalars α_k in Eq. (11), which is essential for each step of the CG method iteration. Since the exact solution $\vec{x}^*$ is unknown, we cannot directly compute the error vector $\vec{e}_{\mathrm{k}} = \vec{x}^* - \vec{x}_{\mathrm{k}}$. However, the residual $\vec{r}_{\mathrm{k}} = \vec{b} - \mathrm{A}\vec{x}_{\mathrm{k}}$ serves as a measurable proxy for the error, representing the negative gradient of the objective function at iteration k. The search direction $\vec{p}_{\mathrm{k}}$, which is constructed to be A-conjugate to direction in the $\mathrm{k}-1$ step, indicates the direction of the k step. α_{k} can be defined using $\vec{r}_{\mathrm{k}}$ and $\vec{p}_{\mathrm{k}}$, as these are the available quantities that provide both geometric (directional) and numerical (magnitude) information required for the update.

This leads to a step size that ensures optimal descent along $\vec{p}_k$ without requiring knowledge of the exact solution.

To ensure that the updated error $\vec{e}_{k+1}$ is A-conjugate to the current direction $\vec{p}_k$, we impose the condition as shown in Figure 10(b):

$$\vec{p}_k^T \mathbf{A} \vec{e}_{k+1} = 0 . \tag{16}$$

Notic that $\mathbf{A}\vec{e}_{k+1} = \mathbf{A}(\vec{x}_{k+1} - \vec{x}^*) = \mathbf{A}\vec{x}_{k+1} - \mathbf{A}\vec{x}^* = -(\vec{b} - \mathbf{A}\vec{x}_{k+1}) = -\vec{r}_{k+1}$. Therefore,

$$\vec{p}_k^T \mathbf{A} \vec{e}_{k+1} = \vec{p}_k^T \vec{r}_{k+1} = 0 . \tag{17}$$

In other words, $\vec{p}_k$ is orthogonal to $\vec{r}_{k+1}$.

Substituting Eq. (14) into Eq. (16):

$$\vec{p}_k^T \mathbf{A}(\vec{e}_k + \alpha_k \vec{p}_k) = 0 , \tag{18}$$

which is simplified to:

$$\vec{p}_k^T \mathbf{A} \vec{e}_k + \vec{p}_k^T \mathbf{A} \alpha_k \vec{p}_k = 0 . \tag{19}$$

Solving for α_k, we obtain:

$$\alpha_k = -\frac{\vec{p}_k^T \mathbf{A} \vec{e}_k}{\vec{p}_k^T \mathbf{A} \vec{p}_k} . \tag{20}$$

Since $\vec{e}_k$ is not directly accessible in practice, we derive another equation for it, by combining $\mathbf{A}\vec{x}^* = \vec{b}$ and $\vec{r}_k = \vec{b} - \mathbf{A}\vec{x}_k$. By eliminating $\vec{b}$:

$$\vec{r}_k = \mathbf{A}(\vec{x}^* - \vec{x}_k) . \tag{21}$$

According to Eq. (12), the error vector is defined as $\vec{e}_k = \vec{x}_k - \vec{x}^*$:

$$\mathbf{A}\vec{e}_k = -\vec{r}_k . \tag{22}$$

Substituted Eq. (22) into Eq. (20) to eliminate $\vec{e}_k$:

$$\alpha_k = \frac{\vec{p}_k^T \vec{r}_k}{\vec{p}_k^T \mathbf{A} \vec{p}_k} . \tag{23}$$

This choice of α_k guarantees that the new point $\vec{x}_{k+1}$ minimizes the function $f(\vec{x})$ along $\vec{p}_k$, while preserving the A-conjugacy of subsequent directions, ensuring rapid and stable convergence. The next task is to determine the search direction $\vec{p}_k$ for the next iteration.

As shown in Figure 11(a), after reaching the new point, we compute the updated residual $\vec{r}_{k+1}$, which points in the direction of the steepest descent. However, we do not want to simply use $\vec{r}_{k+1}$ alone as the next search direction, as this would cause the method to revert to the inefficient steepest descent approach, as shown in the black solid line in Figure 11(b).

Instead, the CG method constructs the new direction as a linear combination of the current residual $\vec{r}_{k+1}$ and the previous direction $\vec{p}_k$ as shown in Figure 11(a):

$$\vec{p}_{k+1} = \vec{r}_{k+1} + \beta_{k+1}\vec{p}_k \,. \tag{24}$$

Once the step along $\vec{p}_k$ is taken, we arrive at a new point where the residual $\vec{r}_{k+1}$ defines the direction of steepest descent, shown as a black dashed arrow pointing upward. Since $\vec{r}_{k+1}$ is orthogonal to $\vec{p}_k$. However, instead of using $\vec{r}_{k+1}$ alone, the CG method adds a scaled component $\beta_{k+1}\vec{p}_k$, depicted as a blue vector pointing in the same general direction as the previous path. Their vector sum forms the new direction $\vec{p}_{k+1}$, shown in red. The scalar β_{k+1} in this process is carefully chosen to ensure that the new direction $\vec{p}_{k+1}$ is conjugate to the previous direction $\vec{p}_k$, resulting in more efficient convergence, as illustrated in the red solid line in Figure 11(b).

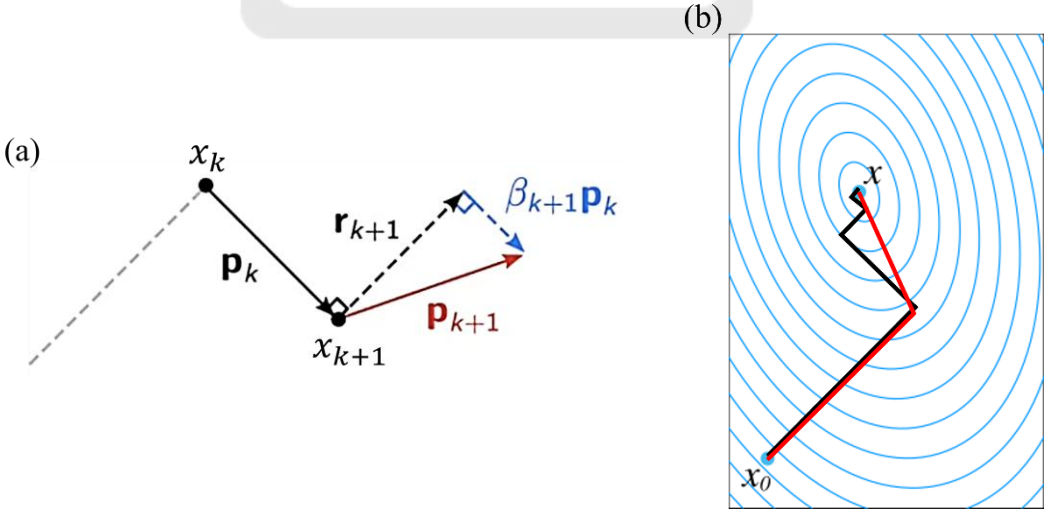

Figure 11: (a) Geometric construction of the new conjugate direction[8]. (b) Conjugate gradient path (red line)[9].

To determine the suitable expressions for the scalars β_{k+1}, we first recall that the residual at any iteration is defined as:

$$\vec{r}_k = \vec{b} - \mathbf{A}\vec{x}_k \,. \tag{25}$$

Using the update rule $\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{p}_k$, we plug this into the residual definition:

$$\vec{r}_{k+1} = \vec{b} - \mathbf{A}\vec{x}_{k+1} = \vec{b} - \mathbf{A}(\vec{x}_k + \alpha_k \vec{p}_k) = \vec{b} - \mathbf{A}\vec{x}_k - \alpha_k \mathbf{A}\vec{p}_k = \vec{r}_k - \alpha_k \mathbf{A}\vec{p}_k \,. \tag{26}$$

In CG method, this new search direction $\vec{p}_{k+1}$ must satisfy two essential conditions: the residual $\vec{r}_{k+1}$ is orthogonal to $\vec{p}_k$, and it must remain A-conjugate to $\vec{p}_k$, to maintain orthogonality under the A-inner product. This requirement can be formally expressed as:

$$\vec{p}_k^{\,T} \mathbf{A} \vec{p}_{k+1} = 0 \,, \tag{27}$$

which prevents the search direction from undoing progress made in previous steps and ensures efficient convergence.

A general form for the new direction is proposed:

$$\vec{p}_{k+1} = \vec{r}_{k+1} + \beta_{k+1} \vec{p}_k \tag{28}$$

as shown in Figure 11(a).

To determine β_{k+1}, we impose conjugation conditions. By substituting Eq. (28) into Eq. (27), we get:

$$\vec{p}_k^{\,T} \mathbf{A}(\vec{r}_{k+1} + \beta_{k+1} \vec{p}_k) = \vec{p}_k^{\,T} \mathbf{A} \vec{r}_{k+1} + \vec{p}_k^{\,T} \mathbf{A} \beta_{k+1} \vec{p}_k = 0 \,. \tag{29}$$

Solving for β_{k+1} gives the exact expression:

$$\beta_{k+1} = -\frac{\vec{p}_k^{\,T} \mathbf{A} \vec{r}_{k+1}}{\vec{p}_k^{\,T} \mathbf{A} \vec{p}_k} \,, \tag{30}$$

which ensures the new direction $\vec{p}_{k+1}$ is A-conjugate to $\vec{p}_k$.

The sequence of using the CG method is:

(1) Start from an initial guess $\vec{x}_k = \vec{x}_0$ for $k = 0$. Define an initial searching vector $\vec{p}_k = \vec{p}_0$ for $k = 0$.

(2) $\vec{r}_k$ is calculated.

(3) α_{k} is calculated using Eq. (23).

(4) $\vec{x}_{\mathrm{k+1}}$ is calculated using Eq. (11).

(5) $\vec{r}_{\mathrm{k+1}}$ is calculated.

(6) Check if $\vec{r}_{\mathrm{k+1}}$ is small enough. If not proceed to the next step, or exit the iteration.

(7) $\beta_{\mathrm{k+1}}$ is calculated using Eq. (30).

(8) $\vec{p}_{\mathrm{k+1}}$ is calculated using Eq. (28).

(9) Go to step (1) with $\vec{x}_{\mathrm{k}} = \vec{x}_{\mathrm{k+1}}$ from step (4) and $\vec{p}_{\mathrm{k}} = \vec{p}_{\mathrm{k+1}}$ from step (8), then repeat step (1) to (9).

Although the conjugate gradient method is obtained above, it involves computing matrix-vector products with $\mathbf{A}$, which can be computationally costly, especially when dealing with large sparse systems. In addition, the first search direction $\vec{p}_0$ is not defined. To improve efficiency, we seek an alternative form of that avoids direct multiplication by $\mathbf{A}$.

To simplify this process and reduce computational complexity, the CG method commonly initializes the first direction using $\vec{p}_0 = \vec{r}_0$, where $\vec{r}_0 = \vec{b} - \mathbf{A}\vec{x}_0$ is the initial residual. This choice is both practical and theoretically justified. In the very first step, the residual $\vec{r}_0$ is the negative gradient of the objective function $f(\vec{x})$. Therefore, choosing $\vec{p}_0 = \vec{r}_0$ means that the initial direction aligns with the direction of steepest descent.

This decision not only simplifies the starting step but also provides structural benefits for subsequent iterations. In particular, the CG method constructs all subsequent directions $\vec{p}_1, \vec{p}_2, \ldots$ in such a way that they are A-conjugate to each other, meaning $\vec{p}_{\mathrm{i}}^{\mathrm{T}}\, \mathbf{A}\vec{p}_{\mathrm{j}} = 0$ for $\mathrm{i} \neq \mathrm{j}$. This orthogonality under the A-inner product is what allows the CG method to reach the exact solution in at most n steps for a n-dimensional system. In fact, this is why the method is called Conjugate Gradient: the directions $\vec{p}_1, \vec{p}_2, \ldots$ are conjugate with respect to the initial gradient direction $\vec{r}_0$.

Moreover, initializing with $\vec{p}_0 = \vec{r}_0$ allows for convenient simplifications in the derivation of both α_{k} and $\beta_{\mathrm{k+1}}$. Because the direction vectors $\vec{p}_{\mathrm{k}}$ are recursively generated using residuals from previous steps, they remain closely aligned, i.e., colinear or in the same subspace, with the corresponding residuals. As a result, Eq. (23) can be simplified to:

$$\alpha_k = \frac{\vec{r}_{\mathrm{k}}^{\mathrm{T}}\vec{r}_{\mathrm{k}}}{\vec{p}_{\mathrm{k}}^{\mathrm{T}}\mathbf{A}\vec{p}_{\mathrm{k}}} . \tag{31}$$

As for the expression of $\beta_{\mathrm{k+1}}$, we take the inner product of both sides of Eq. (26) with $\vec{r}_{\mathrm{k+1}}^{\mathrm{T}}$:

$$\vec{r}_{\mathrm{k+1}}^{\mathrm{T}}\vec{r}_{\mathrm{k+1}} = \vec{r}_{\mathrm{k+1}}^{\mathrm{T}}(\vec{r}_{\mathrm{k}} - \alpha_{\mathrm{k}}\mathbf{A}\vec{p}_{\mathrm{k}}) . \tag{32}$$

Due to the orthogonality condition $\vec{r}_{\mathrm{k+1}}^{\mathrm{T}}\vec{r}_{\mathrm{k}} = 0$:

$$\begin{aligned} \vec{r}_{\mathrm{k+1}}^{\mathrm{T}}\vec{r}_{\mathrm{k+1}} &= -\alpha_k\vec{r}_{\mathrm{k+1}}^{\mathrm{T}}\mathbf{A}\vec{p}_{\mathrm{k}} \\ \Rightarrow \vec{r}_{\mathrm{k+1}}^{\mathrm{T}}\mathbf{A}\vec{p}_{\mathrm{k}} &= -\frac{1}{\alpha_{\mathrm{k}}}\vec{r}_{\mathrm{k+1}}^{\mathrm{T}}\vec{r}_{\mathrm{k+1}} . \end{aligned} \tag{33}$$

Since $\mathbf{A}$ is symmetric, we know $\mathbf{A} = \mathbf{A}^{\mathrm{T}}$. This property allows us to manipulate the inner product involving $\mathbf{A}$ as follows:

$$(\vec{p}_{\mathrm{k}}^{\mathrm{T}}\mathbf{A}\vec{r}_{\mathrm{k+1}})^{\mathrm{T}} = \vec{r}_{\mathrm{k+1}}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\vec{p}_{\mathrm{k}} = \vec{r}_{\mathrm{k+1}}^{\mathrm{T}}\mathbf{A}\vec{p}_{\mathrm{k}} . \tag{34}$$

From Eq. (33), we can get:

$$\vec{p}_{\mathrm{k}}^{\mathrm{T}}\mathbf{A}\vec{r}_{\mathrm{k+1}} = (\vec{p}_{\mathrm{k}}^{\mathrm{T}}\mathbf{A}\vec{r}_{\mathrm{k+1}})^{\mathrm{T}} = \vec{r}_{\mathrm{k+1}}^{\mathrm{T}}\mathbf{A}\vec{p}_{\mathrm{k}} = -\frac{1}{\alpha_{\mathrm{k}}}\vec{r}_{\mathrm{k+1}}^{\mathrm{T}}\vec{r}_{\mathrm{k+1}} . \tag{35}$$

Substituting this into Eq. (30):

$$\beta_{\mathrm{k+1}} = -\frac{-\frac{1}{\alpha_{\mathrm{k}}}\vec{r}_{\mathrm{k+1}}^{\mathrm{T}}\vec{r}_{\mathrm{k+1}}}{\vec{p}_{\mathrm{k}}^{\mathrm{T}}\mathbf{A}\vec{p}_{\mathrm{k}}} = \frac{1}{\alpha_k}\cdot\frac{\vec{r}_{\mathrm{k+1}}^{\mathrm{T}}\vec{r}_{\mathrm{k+1}}}{\vec{p}_k^{\mathrm{T}}\mathbf{A}\vec{p}_{\mathrm{k}}} . \tag{36}$$

Substitute Eq. (31) into Eq. (36), we can derive:

$$\beta_{\mathrm{k+1}} = \frac{\vec{p}_{\mathrm{k}}^{\mathrm{T}}\mathbf{A}\vec{p}_{\mathrm{k}}}{\vec{r}_{\mathrm{k}}^{\mathrm{T}}\vec{r}_{\mathrm{k}}}\cdot\frac{\vec{r}_{\mathrm{k+1}}^{\mathrm{T}}\vec{r}_{\mathrm{k+1}}}{\vec{p}_{\mathrm{k}}^{\mathrm{T}}\mathbf{A}\vec{p}_{\mathrm{k}}} = \frac{\vec{r}_{\mathrm{k+1}}^{\mathrm{T}}\vec{r}_{\mathrm{k+1}}}{\vec{r}_{\mathrm{k}}^{\mathrm{T}}\vec{r}_{\mathrm{k}}} . \tag{37}$$

At this point, we have obtained all the necessary quantities for performing the CG method, including the step size α_{k}, the new residual $\vec{r}_{\mathrm{k+1}}$, the direction update factor β_{k+1}, and

the new search direction $\vec{p}_{k+1}$. These quantities can all be computed using inner products and matrix-vector multiplications, without the need to store or invert the matrix $\mathbf{A}$, which makes CG method especially efficient for large, sparse, symmetric positive-definite systems.

The full CG method proceeds starting from an initial guess $\vec{x}_0$, we compute the initial residual $\vec{r}_0 = \vec{b} - \mathbf{A}\vec{x}_0$ and set the first direction as $\vec{p}_0 = \vec{r}_0$. In each iteration, we calculate the optimal step size:

$$\alpha_k = \frac{\vec{r}_{\mathrm{k}}^{\mathrm{T}}\vec{r}_{\mathrm{k}}}{\vec{p}_{\mathrm{k}}^{\mathrm{T}}\mathbf{A}\vec{p}_{\mathrm{k}}} \tag{38}$$

to update the solution

$$\vec{x}_{\mathrm{k+1}} = \vec{x}_{\mathrm{k}} + \alpha_{\mathrm{k}}\vec{p}_{\mathrm{k}}\,. \tag{39}$$

Next, we compute the new residual:

$$\vec{r}_{\mathrm{k+1}} = \vec{b} - \mathbf{A}\vec{x}_{\mathrm{k+1}}\,. \tag{40}$$

Then, we construct the new search direction using the formula:

$$\beta_{k+1} = \frac{\vec{r}_{\mathrm{k+1}}^{\mathrm{T}}\vec{r}_{\mathrm{k+1}}}{\vec{r}_{\mathrm{k}}^{\mathrm{T}}\vec{r}_{\mathrm{k}}}, \quad \vec{p}_{\mathrm{k+1}} = \vec{r}_{\mathrm{k+1}} + \beta_{k+1}\vec{p}_{\mathrm{k}}\,. \tag{41}$$

This process repeats until the residual norm $||\vec{r}_{\mathrm{k+1}}||$ falls below a specified tolerance.

The geometric idea behind this method is clearly illustrated in Figure 8. Starting from the point $\vec{x}_{\mathrm{k}}$, we move along the search direction $\vec{p}_{\mathrm{k}}$ by a distance α_k arriving at a new position $\vec{x}_{\mathrm{k+1}}$. From there, we calculate the next search direction $\vec{p}_{\mathrm{k+1}}$ which lies in a different direction but still ensures conjugacy with respect to the previous directions. The solid arrow from $\vec{x}_{\mathrm{k}}$ to $\vec{x}_{\mathrm{k+1}}$ represents the step $\alpha_k\vec{p}_{\mathrm{k}}$, and the dashed arrow indicates the new direction $\vec{p}_{\mathrm{k+1}}$. By combining information from both the current residual and the previous search direction, this update ensures both efficient descent and mathematical orthogonality, driving convergence to the true solution $\vec{x}^*$.

3.3. Components defined in the calculation

mini-Tokamak primarily relies on the time-varying current in the central solenoid (indicated by the blue arrow in Figure 12) for plasma generation and driving plasma currents (represented by the orange arrow in Figure 12). Consequently, the central solenoid undergoes significant current variations that induce currents in the surrounding components, e.g. the vacuum-vessel walls. As the vacuum-vessel walls are the primary components affected and the main carrier of eddy currents, we first focus on calculating the eddy currents in the inner and outer vacuum-vessel walls (depicted by the green and yellow arrows, respectively, in Figure 12) generated by the time-varying current of the central solenoid.

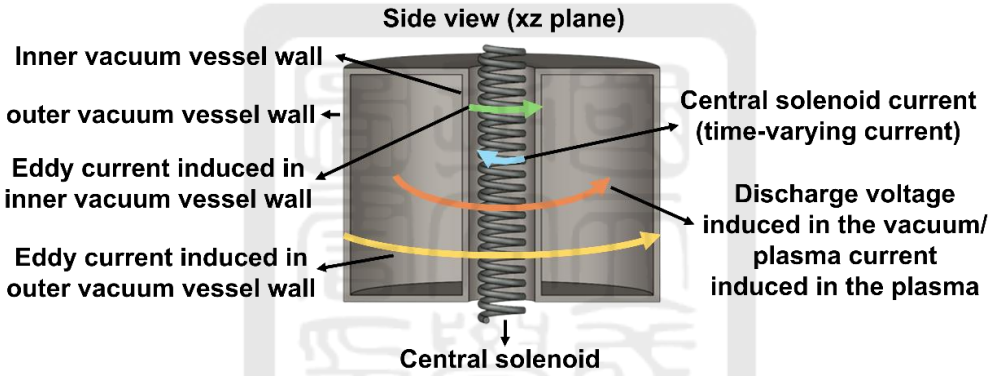

Figure 12: Schematic diagram of currents in the mini-Tokamak.

As shown in Figure 13(a), the central solenoid consists of two layers, each containing 92 turns, for a total of 184 turns. The distances from the centerline to the centers of the inner and outer layers are 8 mm and 14 mm, respectively. For simplicity, the major radius of the central solenoid in our calculations is set as their average, 11 mm. The solenoid coil is a solid copper wire wrapped with an insulating layer on the outside. The blue circle represents the insulation layer, which is 1.7 mm thick. The black circle represents the coil, with a diameter of 2.6 mm.

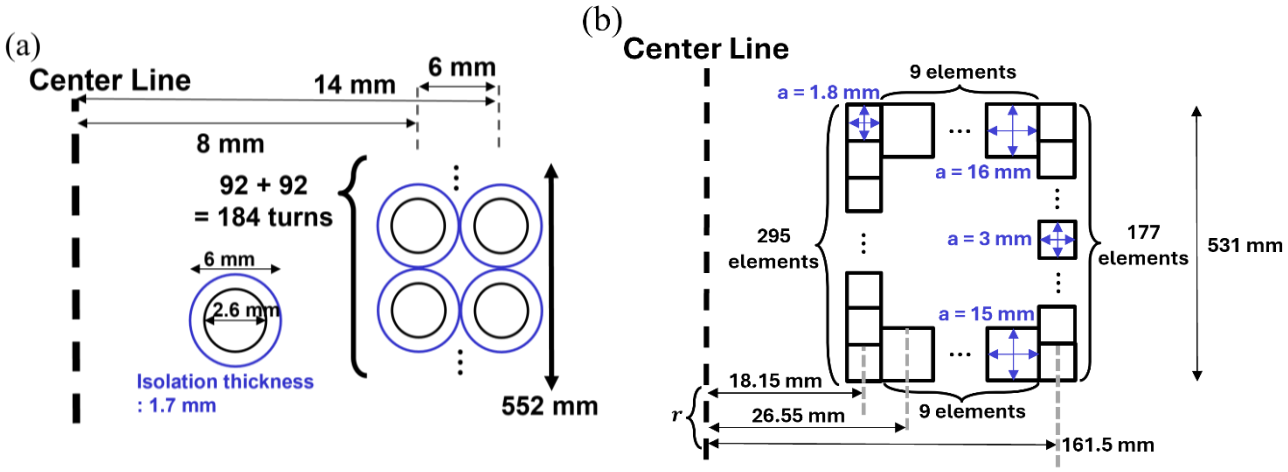

Figure 13: (a) Specifications of the central solenoid in mini-Tokamak. (b) Specifications of the inner and outer vacuum-vessel walls divided for calculation.

For the vacuum vessel, we divide it into four parts: the inner-vacuum-vessel wall and bottom lid, and the outer-vacuum-vessel wall and top lid, as illustrated in Figure 13(b). The vacuum-vessel wall is treated as many ring elements stacking on top of or beside each other. To enhance calculation accuracy, as illustrated in Figure 13(b), the inner-vacuum-vessel wall is divided into 295 identical ring elements with square cross-sections of 1.8 mm on each side, stacked together. Similarly, the outer-vacuum-vessel wall is divided into 177 ring elements with square cross-sections of 3 mm on each side. For the top and bottom lids, although they have the same axial length, their thicknesses differ slightly, 15 mm for the bottom lid and 16 mm for the lid. Due to rounding, both lids are segmented into 9 identical ring coils, each with square cross-sections of 15 mm × 15 mm and 16 mm × 16 mm, respectively. This segmentation allows for precise modeling of the eddy currents generated within the vacuum-vessel walls.

All components are implemented in MATLAB according to the specifications, and the detailed code is provided in Appendix A.4. Currently, we only consider the central solenoid and the vacuum-vessel wall (referred to as the "chamber" in the code). To include

the plasma or poloidal field coils (PFCs) in the calculation, one simply needs to change the corresponding flag from false to true at the beginning of the script.

3.4. Calculation of inductance and resistance of each component

Before calculating the induced currents, we first need to compute the resistance and self-inductance of the central solenoid and vacuum-vessel walls, as well as the mutual inductance between them.

This section consists of three parts. Section 3.4.1 shows the calculation of the resistance of each component. Section 3.4.2 presents the calculation of self-inductance of each component, and Section 3.4.3 demonstrates the calculation of mutual inductance between the components.

3.4.1. Calculation of resistance

We use the resistance formula to calculate the resistance of the components:

$$R = \eta \frac{L}{A} \tag{42}$$

where R is the coil resistance, η is the resistivity of the material, L is the length of the coil, A is the cross-sectional area of the coil.

(a) The central solenoid

We first divide the central solenoid into inner and outer layers and treat each turn within each layer as a single ring coil:

$$R_{\mathrm{cs\_each}} = \eta_{\mathrm{copper}} \frac{L_{\mathrm{cs\_each}}}{A_{\mathrm{cs\_each}}} = \eta_{\mathrm{copper}} \frac{2\pi r_{\mathrm{cs\_each}}}{\pi a_{\mathrm{cs\_each}}^2} \tag{43}$$

where $R_{\mathrm{cs\_each}}$ is the resistance of a single coil in the inner or outer layer, η_{copper} is the resistivity of copper $(1.68 \times 10^{-8}\ \Omega \cdot \mathrm{m})$, $A_{\mathrm{cs\_each}}$ is the cross-sectional area of the coil, and $2\pi r_{\mathrm{cs\_each}}$ is the coil's circumference, where $r_{\mathrm{cs\_each}}$ being the major radius of the coil. For all inner layer coils, $r_{\mathrm{inner\_cs\_each}} = 8\ \mathrm{mm}$, and for all outer layer coils, $r_{\mathrm{outer\_cs\_each}} = 14\ \mathrm{mm}$, as shown in Figure 13(a). The term $\pi a_{\mathrm{cs\_each}}^2$

represents the cross-sectional area of the coil, where $a_{\text{cs\_each}}$ is the radius of the coil. For all inner and outer layer coils, $a_{\text{cs\_each}} = 1.3$ mm, as shown in Figure 13(a).

After obtaining the resistance of each single coil in each layer, we can calculate the total resistance of the central solenoid (R_{cs}) by summing them up:

$$\sum R_{\text{inner\_cs\_each}} + \sum R_{\text{outer\_cs\_each}} = R_{\text{cs}} . \tag{44}$$

(b) The vacuum-vessel walls

We calculate the resistance of each element of the vacuum-vessel walls in Figure 13(b) separately:

$$R_{\text{vessel\_each}} = \eta_{\text{stainless\_steel}} \frac{L_{\text{vessel\_each}}}{A_{\text{vessel\_each}}} = \eta_{\text{stainless\_steel}} \frac{2\pi r_{\text{vessel\_each}}}{a^2_{\text{vessel\_each}}} \tag{45}$$

where $R_{\text{vessel\_each}}$ is the resistance of an element of in the vacuum-vessel walls, $\eta_{\text{stainless\_steel}}$ is the resistivity of stainless steel $(7 \times 10^{-7}\ \Omega \cdot \text{m})$, $A_{\text{vessel\_each}}$ is the cross-sectional area of the element, and $2\pi r_{\text{vessel\_each}}$ is the element 's circumference, where $r_{\text{vessel\_each}}$ is the radius of the element. For all elements of inner-vacuum-vessel wall elements, $r_{\text{vessel\_each}} = 18.15$ mm. For all elements of outer-vacuum-vessel wall elements, $r_{\text{vessel\_each}} = 161.5$ mm. As for the bottom lid, $r_{\text{vessel\_each}}$ increases linearly from 26.55 mm to 146.55 mm in 15 mm increments. For the top lid, $r_{\text{vessel\_each}}$ decreases linearly from 152 mm to 24 mm in 16 mm increments. This spatial arrangement is shown in Figure 13(b). The term $a^2_{\text{vessel\_each}}$ represents the cross-sectional area of the element, where $a_{\text{vessel\_each}}$ is the side length of the square element. For all inner-vacuum-vessel walls elements, $a_{\text{vessel\_each}} = 1.8$ mm; for all the outer vacuum-vessel walls elements, $a_{\text{vessel\_each}} = 3$ mm; for all bottom lid elements, $a_{\text{vessel\_each}} = 15$ mm; for all top lid elements, $a_{\text{vessel\_each}} = 16$ mm; as shown in Figure 13(b).

Figure 14 illustrates the calculated resistance of each component. The first isolated point represents the resistance of the central solenoid, while component numbered 2 to 296 correspond to the inner-vacuum-vessel wall ring elements shown in Figure 13(b), starting from the bottommost element and sequentially moving upward to the 296th element. Since the inner-vacuum-vessel wall ring elements are identical in geometry, except for location in z, their resistance remains the same.

Component numbered 297 to 305 correspond to the bottom lid wall ring elements in Figure 13(b). The resistance of these elements increases as the major radius increases. Upon reaching the outer-vacuum-vessel wall, corresponding to components 306 to 482, the major radius remains constant while only the vertical position varies, resulting in a flat resistance profile. At the top lid, corresponding to components 483 to 491, the major radius begins to decrease inward, leading to a corresponding decrease in resistance. Due to the geometric symmetry about $z = 0$, the upper and lower halves of the outer vacuum-vessel wall exhibit symmetric resistance behavior. This symmetry is evident at the midpoint coil (coil number 392), located approximately at $z = 0$, where the resistance values on either side are nearly identical.

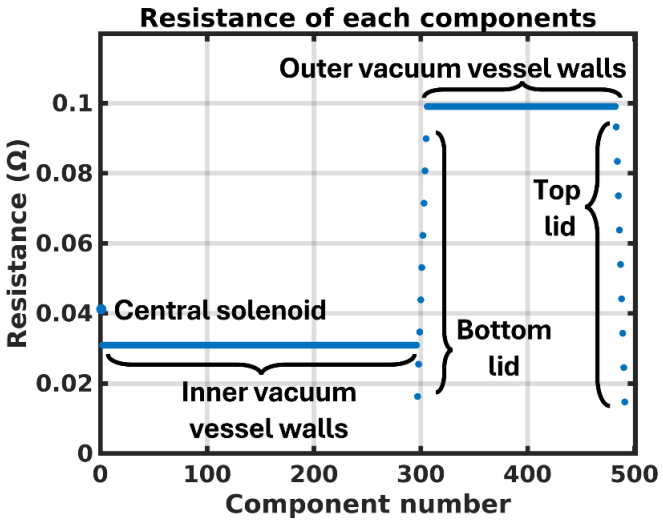

Figure 14: Resistance of each component.

3.4.2. Calculation of self-inductance

In our calculations of inductance, all equations are from the book *Inductance Calculations: Working Formulas and Tables* by Frederick Grover[10]. One important point to note is that the calculations in the book are performed using units of centimeters, and the results obtained directly from the equations are in microhenries (μH). Therefore, it is crucial to ensure that the input units used in the equations and the output units from the results are consistent and properly accounted for.

(a) The central solenoid

We use the equation of solenoid self-inductance for the calculation[12]:

$$L_{\mathrm{cs}} = \mu_0 (\frac{N}{l})^2 Al = \mu_0 \frac{N^2}{l} A = \mu_0 \frac{N^2}{l} \pi r^2 \tag{46}$$

where L_{cs} is the self-inductance of central solenoid, μ_0 is the permeability of free space $(4\pi \times 10^{-7}\ {}^{H}/_{m})$, $N = 184$ is the total number of turns in the solenoid, $l = 552$ mm is the height of the solenoid, A is the cross-sectional area of the central solenoid, and $r = 11$ mm is the average major radius of the central solenoid as mentioned in section 3.3 and illustrated in Figure 13(a).

(b) The vacuum-vessel walls

The self-inductance of each element can be calculated using the formula for the inductance of circular coils with square cross-sections, as given in Eq. 91 on page 95 of Ref. [10]:

$$L = 0.001 a N^2 P_0' \ \mu H \tag{47}$$

where L is the element's self-inductance, and N is the number of turns in the element. In our calculation, $N = 1$ since all elements were represented as single coils. P_0' is a variable that can be calculated using Eq. 92 on page 95 of Ref. [10]:

$$P_0' = 4\pi \left\{ \frac{1}{2}\left[1 + \frac{1}{6}\left(\frac{a}{2r}\right)^2\right] log_e \left[\frac{8}{\left(\frac{a}{2r}\right)^2}\right] - 0.84834 + 0.2041\left(\frac{a}{2r}\right)^2 \right\} \quad (48)$$

where r is the major radius of the element, and a is the side length of the element.

Figure 15 illustrates the self-inductance of each component. The first isolated point represents the self-inductance of the central solenoid, which has the highest inductance among all components due to its large number of turns and compact geometry. The self-inductance of component numbered 2 to 296, corresponding to the inner-vacuum-vessel wall ring elements shown in Figure 13(b). These coils are arranged sequentially from the bottommost coil to the 296th coil. Since the inner-vacuum-vessel wall ring elements are geometrically identical except for their z-coordinate, their self-inductance values are the same.

For component numbered 297 to 305, correspond to the bottom lid wall ring elements, the self-inductance varies primarily with the coil radius. As shown in Figure 15, the self-inductance increases as the major radius increases from the bottom lid outward. Once reach the outer-vacuum-vessel wall ring elements, corresponding to components 306 to 482, the major radius remains constant while only the vertical position changes, resulting in a

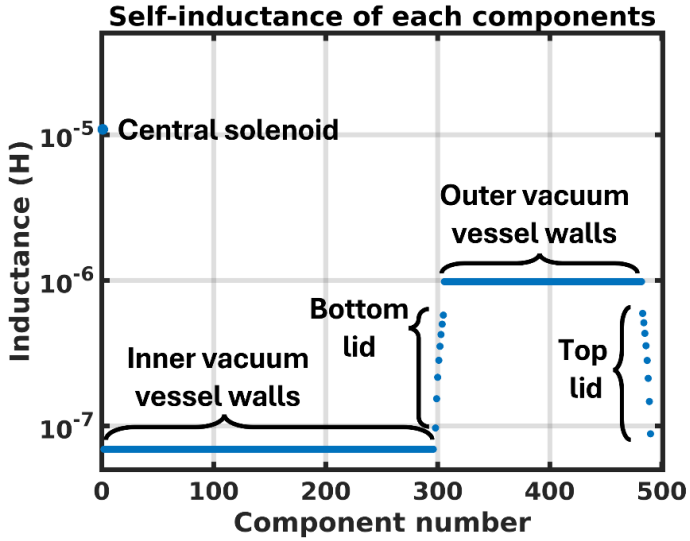

Figure 15: Self-inductance of each component.

plateau in self-inductance. As the coil sequence continues toward the top lid, corresponding to components 483 to 491, the major radius decreases inward, causing the self-inductance to decrease accordingly. Due to the symmetry about $z = 0$, the self-inductance profile is nearly symmetric with respect to the middle coil (coil number 392), which lies close to $z = 0$.

3.4.3. Calculation of mutual inductance

Our calculation consider all components, except for the central solenoid, as ring coils. Therefore, we calculated: (1) mutual inductance between elements of vacuum-vessel wall and the central solenoid; (2) mutual inductance between the elements of vacuum-vessel wall.

3.4.3.1. Mutual inductance between elements of vacuum-vessel wall and the central solenoid

Due to different vertical locations of elements of the vacuum-vessel wall, we categorize them into two groups. The first group consists of elements located between the top and bottom planes of the central solenoid so that the central solenoid is divided into two parts: the top solenoid and the bottom solenoid, as shown in Figure 16(a). The second group consists of elements located either above the top plane or below the bottom plane of the central solenoid, as illustrated in Figure 16(b), which is the case for elements below the bottom plane of the central solenoid as an example. For these two different cases, we employ different calculation methods for the mutual inductance between the central solenoid and the vacuum-vessel walls.

(1) Elements between the top plane and the bottom plane of the central solenoid

When the single element is positioned between the top plane and the bottom plane of the central solenoid, as shown in Figure 16(a), the central solenoid can be divided into two parts: the portion of the solenoid above the element's plane (referred to as the top solenoid) with a length of x and the portion below the element's plane (referred to as the bottom solenoid) with a length of $(l - x)$, where l is the length of the central solenoid. We can calculate the mutual inductance between the single element and the top solenoid, as well as the mutual inductance between the single element and the bottom solenoid. Finally, the sum of these two mutual inductances gives the total mutual inductance between the central solenoid and the element.

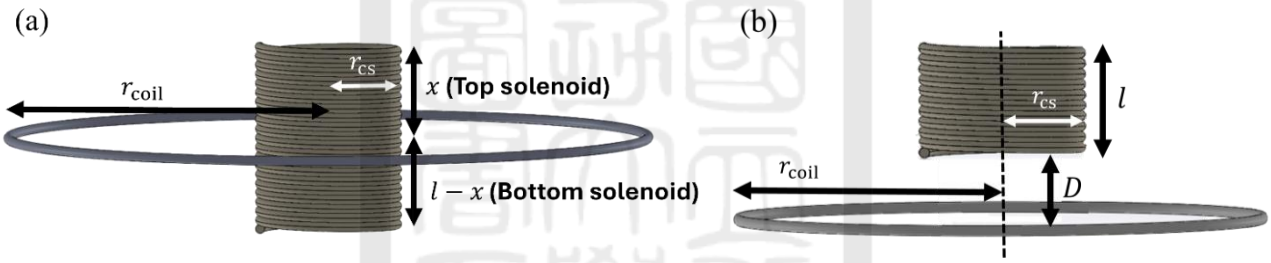

Figure 16: (a) Schematic diagram of coil located within the range of the solenoid.
(b) Schematic diagram of coil located outside the range of the solenoid.

Elements of vacuum-vessel walls are single coils located coaxially with the central solenoid, Therefore, the mutual inductance between a solenoid and the element located at its end plane can be calculated using the formula from Eq. 103 on page 115 of Ref [10]:

$$M_{\mathrm{T}/B} = 0.002\pi^2 r_{\mathrm{cs}} \alpha \rho N Q_0 \tag{49}$$

where $M_{\mathrm{T/B}}$ is the mutual inductance between the top/bottom solenoid and a coaxial single coil at its end plane, as illustrated in Figure 17. Figure 17(a) is the case for the top solenoid while Figure 17(b) is the case for bottom solenoid. In our calculation, $N = 184$ is the number of turns in the top or the bottom solenoid. Q_0 can be obtained from Table 27 in

page 115 of Ref [10], as shown in Appendix A.5. r_{cs} is the major radius of the solenoid, which is 11 mm, as mentioned in section 3.3. The variable α can be calculated using the following equation:

$$\alpha = \frac{r_{\mathrm{cs}}}{r_{\mathrm{coil}}} \tag{50}$$

where r_{coil} is the major radius of the coil in Figure 17(a) and (b), corresponding to the r in Figure 13(b) for each component of the vacuum-vessel wall. The variable ρ can be calculated by:

$$\rho = \sqrt{\frac{{r_{\mathrm{coil}}}^2}{{r_{\mathrm{coil}}}^2 + s^2}} \tag{51}$$

where $s = x$ or $s = l - x$ is the distance between the top or the bottom of the central solenoid and the elements plane, respectively, as shown in Figure 17(a) and (b). Finally, the mutual inductance between the element and the central solenoid is obtained:

$$M = M_{\mathrm{T}} + M_{\mathrm{B}}, \tag{52}$$

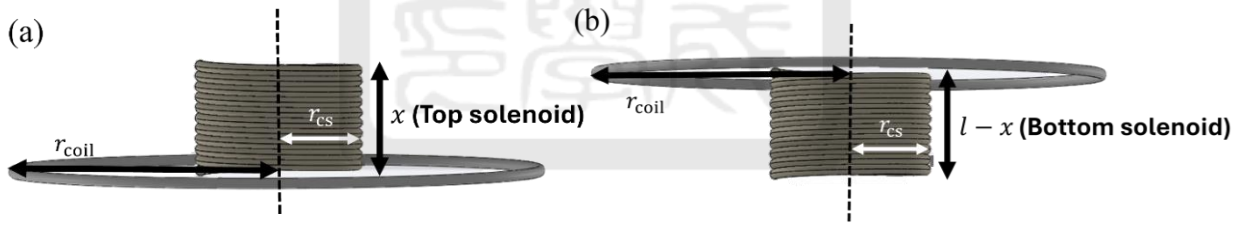

Figure 17: (a) Coil schematic diagram for top solenoid. (b) Coil schematic diagram for bottom solenoid.

(2) Mutual inductance between the central solenoid and the elements with a distance D away from the end of the central solenoid

When calculating the mutual inductance between the central solenoid and the element with a distance, we can imagine there is a solenoid between the central solenoid and the element, as illustrated by the light-colored solenoid in Figure 18. We first calculate the

mutual inductance M_{l+D} between the "solenoid $l+D$" with a length of $l+D$, and the element using Eq. (49). Then we calculate the mutual inductance M_D of the imaginated "solenoid D" with a length of D, and the element using Eq. (49) and get M_{D}. Finally, we can obtain the mutual inductance between the central solenoid and the element at any distance away from the central solenoid by:

$$M_{\mathrm{x}} = M_{\mathrm{x+D}} - M_{\mathrm{D}}, \tag{53}$$

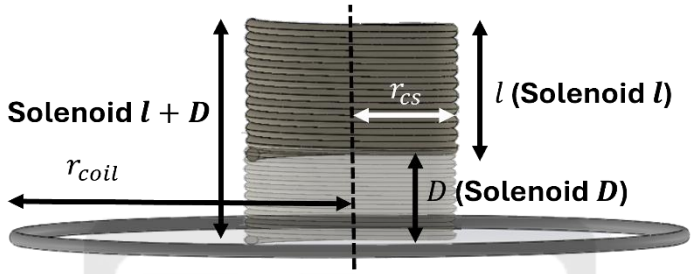

Figure 18: Schematic diagram of coils located outside the range of the solenoid with the imagined solenoid.

3.4.3.2. Mutual inductance between elements of the vacuum-vessel walls

The mutual inductance between elements of vacuum-vessel walls, including the inner and outer vacuum-vessel walls, and the top and bottom lids, can be calculated using the formula for the mutual inductance of coaxial circular filaments, as shown in Figure 19, which is given in Eq. 77 on page 77 of Ref [10]:

$$M = f\sqrt{r_{\mathrm{c1}} r_{\mathrm{c2}}} \tag{54}$$

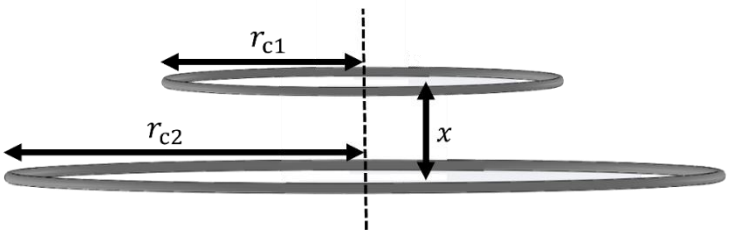

Figure 19: Coil schematic diagram of two ring coils.

where M represents the mutual inductance of coaxial circular filaments. The value of f can be obtained from Table 13 to Table 17 of Ref [10], corresponding to different ratios of r_{c1} and r_{c2} as shown in Appendix A.6 to A.10. It is a function of the distance x between two rings, as well as their radius r_{c1} and r_{c2}. r_{c1} is the major radius of the smaller single coil, and r_{c2} is the major radius of the larger single coil, as shown in Figure 19.

3.4.3.3. Calculation results of mutual inductance

Figure 20, as an example, illustrates the mutual inductance between all components and the 88th outer-vacuum-vessel wall element (392th component in Figure 20), which corresponds to the middle outer-vacuum-vessel wall element depicted in Figure 13(b).

The mutual inductance values depend on the spatial relationship between components. The first data point represents the mutual inductance between the central solenoid and the 392th component. For component numbered 2 to 296, corresponding to elements of the inner-vacuum-vessel wall, the mutual inductance gradually increases as the z-coordinates of these elements approach the plane of the 392th component and decrease as the elements move away from the 392th component.

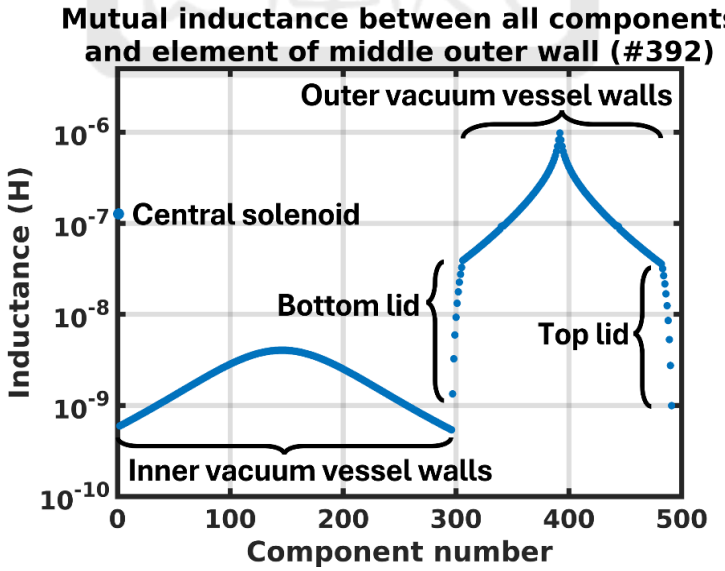

Figure 20: Mutual inductance between all components and the 88th outer vacuum-vessel wall ring coil.

For component numbered 296 to 305, representing the bottom lid elements, the mutual inductance increases as the z-coordinates of these elements approach the plane of the 392th component. Once reach the outer-vacuum-vessel wall ring elements, corresponding to components 306 to 482, the mutual inductance keeps increasing as the z-coordinates of these elements approach the plane of the 392th component and peaks at the 392th ring coil itself, which is nearly equal to its self-inductance. Beyond this point, the mutual inductance decreases for elements located farther from the 392th coil, including the top lid for component numbered 483 to 491, due to increasing spatial separation, as illustrated in Figure 20.

Using Eq. (54), we can calculate not only the mutual inductance between each element of the inner and the outer vacuum-vessel wall, but also the mutual inductance between vacuum-vessel elements and other coils, such as poloidal field coils, or the mutual inductance between two poloidal field coils, provided that the two single coils are coaxial.

Using Eq. (49) to (54), we can calculate the mutual inductance between the central solenoid and any vacuum-vessel wall element, as well as the mutual inductance between vacuum-vessel wall elements.

Appendix A.11 provides the detailed code for resistance and inductance calculations. At present, the model includes only the central solenoid and the vacuum-vessel wall ("chamber"). Plasma can be included by setting the relevant flag from false to true at the start of the script.

3.5. Calculated results of the eddy currents in the vacuum-vessel walls

In our current calculation, we aim to determine the eddy currents in the inner and the outer vacuum-vessel walls induced by the time-varying current of the central solenoid by solving Eq. (5) using the CG method[11] introduced in Section 3.2 with the resistance and the inductance of components defined in Section 3.3 and calculated in Section 3.4. The calculation involves 295 ring elements in the inner-vacuum-vessel wall, and 195 ring

elements in the outer-vacuum-vessel wall, which we include the top and bottom lids from now, i.e., a total of 490 vacuum-vessel elements. Including the central solenoid, there are 491 components in total. The detailed code for eddy current calculation is provided in Appendix A.12.

The calculated result $\vec{I}_{\mathrm{cv}}$ will be represented as a column vector with a size of 491, as shown in Eq. (55). I_{c1} represents the time-varying current for central solenoid. I_{v1} to I_{v490} represent the currents in the vacuum-vessel elements, which correspond to the eddy currents we aim to calculate:

$$\vec{I}_{\mathrm{cv}} = \begin{bmatrix} I_{\mathrm{cs}} \\ I_{\mathrm{v1}} \\ \vdots \\ I_{\mathrm{v490}} \end{bmatrix}. \tag{55}$$

Notice that I_{cs} is given while I_{v1} to I_{v490} are eddy currents we obtain.

With **M** and **R** calculated in Section 3.4, the eddy currents induced in each ring element of the inner and outer vacuum-vessel walls can be determined based on the time-varying current I_{cs} in the central solenoid. Figure 21 is an example illustrating the calculated results. The current profile of the solenoid, shown by the red curve in Figure 21(a), begins with a linear ramp-up from 0 to 4 kA over the first 20 ms, followed by a flat-top phase maintaining 4 kA for the next 20 ms. Subsequently, the current linearly ramps down from 4 kA to 0 within 15 ms, resulting in a total duration of 55 ms.

The calculated results of the eddy currents distribution are presented in Figure 21(b) and (c). Since the inner and outer vacuum-vessel walls are divided into several ring elements, Figure 21(b) shows 295 eddy current lines corresponding to the 295 ring elements of the inner-vacuum-vessel wall. Similarly, Figure 21(c) shows 195 eddy current lines corresponding to the 195 ring elements of the outer-vacuum-vessel wall.

By summing up the eddy currents from all 295 elements of the inner-vacuum-vessel wall, we can calculate the total eddy current generated in the entire inner-vacuum-vessel wall, represented as the blue solid line in Figure 21(a). Similarly, by summing up the eddy currents from all 195 elements of outer-vacuum-vessel wall, we can calculate the total eddy current generated in the entire outer-vacuum-vessel wall, represented as the blue dashed line in Figure 21(a).

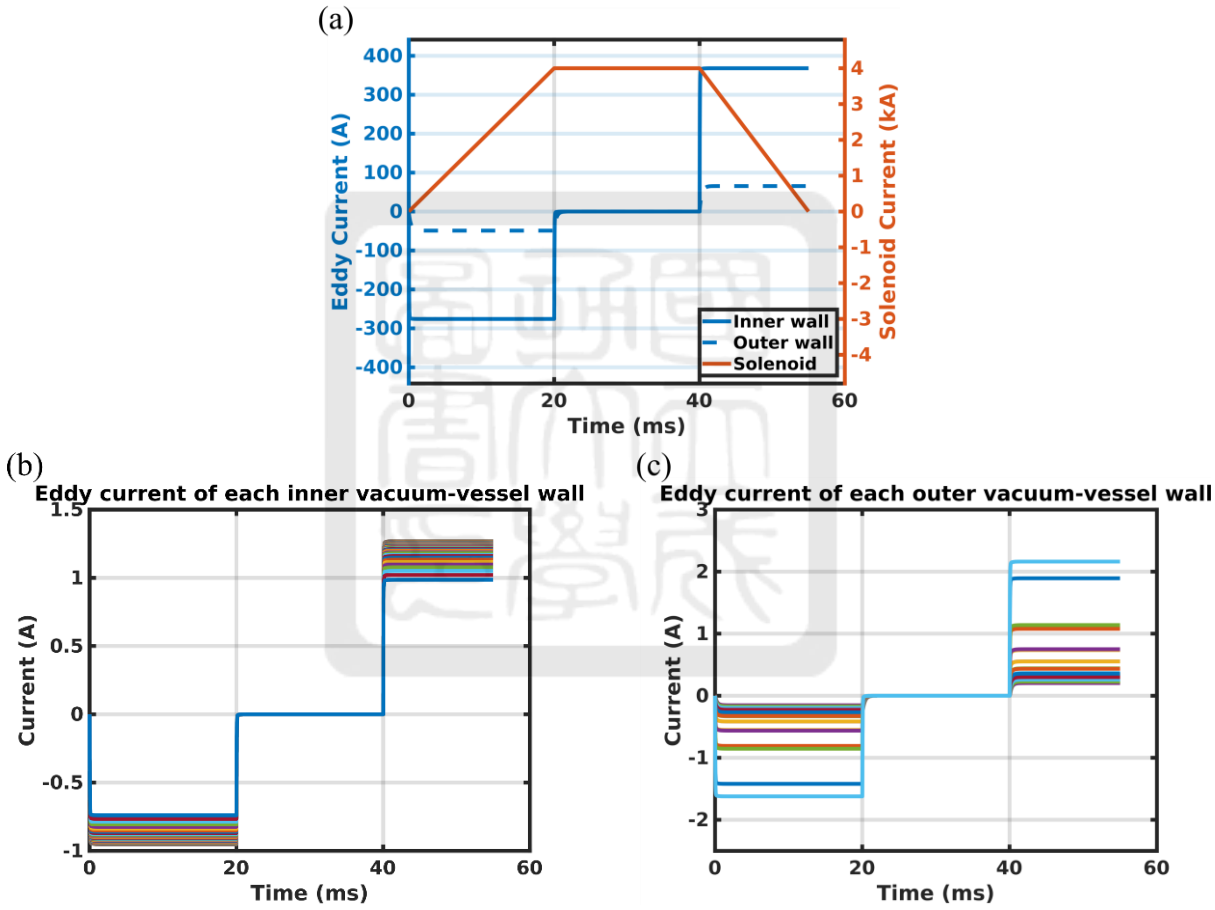

Figure 21: (a) Current profile of central solenoid and the eddy currents of the inner and outer vacuum-vessel wall. (b) The eddy current of each element of the inner-vacuum-vessel wall. (c) The eddy current of each element of the outer-vacuum-vessel wall.

3.6. Conclusion

We can now calculate the eddy currents in the inner and outer vacuum-vessel walls induced by the time-varying current of the central solenoid using Conjugate Gradient (CG) method. This method accelerates convergence by reducing the number of iterations needed while maintaining numerical stability. This is crucial as our matrix size increases with more calculated components, ensuring the computation remains scalable and efficient. This allows us to incorporate the influence of eddy currents into our subsequent design calculations to achieve more accurate results.

4. Calculation of required loop voltage for breakdown

mini-Tokamak employs a central solenoid (CS) driven by a time-varying current to produce a time-varying magnetic field which induces a loop voltage. The loop voltage can be used to generate the plasma or drive the plasma current. If it exceeds the breakdown voltage, the gas is ionized, and plasma is generated. The breakdown voltage is influenced by factors such as background pressure, connection length, and the plasma's position within the tokamak. With these factors, the loop voltage and the time-varying current profile can be determined.

The purpose of this section is to develop an algorithm to calculate the required loop voltage for breakdown under varying conditions and determine the required central solenoid current change rate to generate that loop voltage. It is to ensure that the central solenoid and the current profile that drives the central solenoid can meet the plasma generation demands, providing a critical basis for validating and optimizing the design. In addition, it helps identify potential design limitations and guides adjustments to improve the central solenoid's overall design. This chapter consists of six sections. Section 4.1 explains the breakdown voltage required for plasma generation. Section 4.2 presents the required rate of change of the central solenoid current to induce sufficient loop voltage for initiating breakdown. Section 4.3 calculates the eddy currents induced by this current profile, while Section 4.4 shows the exact loop voltage induced by the same current profile, including the effect of eddy currents. Section 4.5 discusses the limitations and assumptions, and Section 4.6 provides the conclusion.

4.1. Breakdown voltage for plasma generation

As shown in Figure 22, plasma breakdown in a DC discharge occurs when the applied voltage across the anode (positive electrode) and the cathode (negative electrode) becomes sufficient to ionize the neutral gas molecules between the electrodes. The process begins with

an initial ionization event near the cathode, where a free electron is liberated. This electron is accelerated by the electric field. If it has enough kinetic energy and collides with a neutral gas molecule, it may ionize the neutral gas molecule and release an additional free electron. Both electrons can be accelerated by the electric field and generate more electrons through the same process. All electrons including the original electron and the newly liberated electron continue the process, leading to an exponential increase in the number of charged particles. The blue lines in the figure represent the paths of ionizing electrons, while the orange lines represent the paths of liberated electrons. This cascading ionization process is known as the Townsend avalanche.

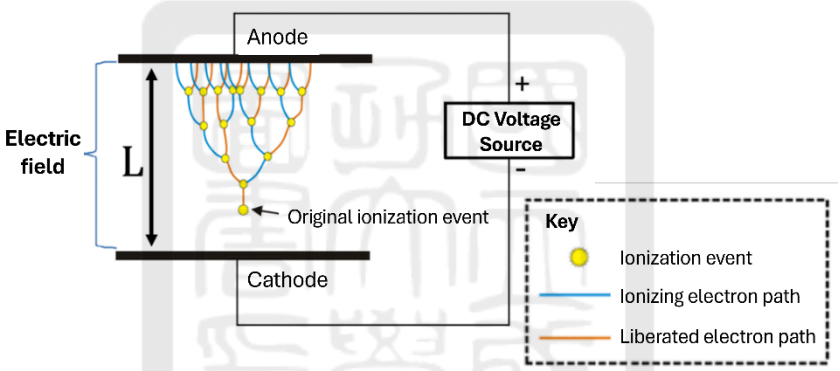

Figure 22: Visualization of Townsend avalanche[13].

The connection length (L), marked by the vertical distance between the anode and cathode in Figure 22, defines the region within which the electric field acts to sustain the avalanche. This length directly impacts the number of ionization events that can occur before the electrons reach the anode. A longer connection length provides more chance for the avalanche to develop, increasing the total number of electrons and ions generated. Conversely, a shorter connection length limits the ionization path, reducing the amount of ionization.

In contrast, there is no anode and cathode in the tokamak system to provide the electric field. Alternatively, loop voltage in the azimuthal direction, i.e., an azimuthal electric field,

is generated by the central solenoid with time-varying current. Electrons gain energy from the loop voltage, in the tokamak setup illustrated in Figure 23. On the other hand, the presence of an external toroidal magnetic field, applied by the toroidal magnetic field coils, determines the connection length. Unlike the finite connection length in a DC discharge shown in Figure 22, the tokamak introduces magnetic field lines (represented by the blue solid line in Figure 23) that forces electrons to gyro around field lines and move freely along them. Instead of being limited by a physical distance between electrodes, the connection length in the tokamak becomes theoretically infinite because the electrons follow the closed magnetic field lines. As a result, in the tokamak, electrons follow toroidal magnetic field lines and collide with neutral gas molecules. This process ionizes the neutral gas and releases additional free electrons, sustaining a continuous avalanche.

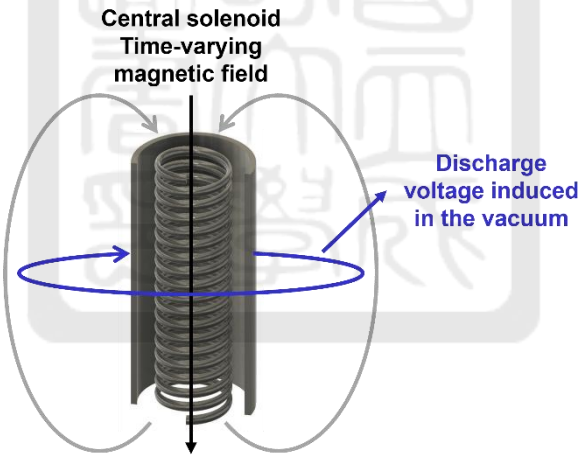

Figure 23: Stray magnetic field in a tokamak.

In the tokamak, we utilize the central solenoid to induce the discharge voltage, also known as the loop voltage, as shown in Figure 23. For an ideal solenoid, its magnetic field should be uniform in the solenoid and aligned along its central axis, as represented by the black solid line pointing downward in the middle of the solenoid in Figure 23. In reality, the solenoid produces stray magnetic field lines, as shown by the gray solid lines in Figure 23.

The magnetic field lines become helical when the stray magnetic field combines with the toroidal field. Instead of moving azimuthally along the toroidal magnetic field, electrons follow a helical path and move upward, as represented by the black solid line in Figure 24. Eventually, electrons collide with the outer-vacuum-vessel wall and are lost. This results in a finite connection length, in contrast to the infinite connection length observed in the ideal scenario.

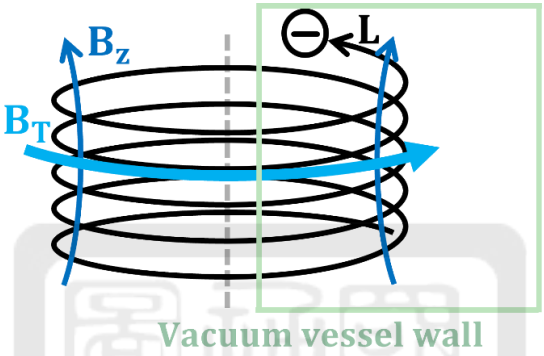

Figure 24: Electron path in a tokamak.

To calculate the connection length in our tokamak, we can straighten the black helical curve in Figure 24, as represented by the hypotenuse of the triangle, side c in Figure 25. The vertical side of the triangle, side b in Figure 25, corresponds to the electron's displacement in the z-direction in Figure 24. Assuming the electron originates at a point 85 mm from the centerline at the bottom of the vacuum vessel, which corresponds to the major radius of the plasma in the mini-Tokamak described in Chapter 2, and maintains the same distance from the centerline (85 mm) throughout its upward displacement, as shown by the black line in

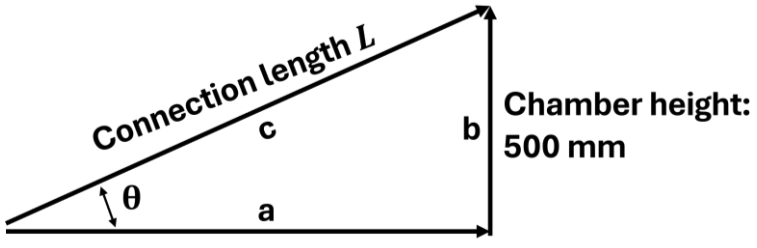

Figure 25: Electron path in a tokamak.

Figure 24, it eventually moves to the top of the vacuum vessel and is lost. In this case, since the internal height of the vacuum vessel can be obtained by subtracting the width of the top (16 mm) and bottom (15 mm) lids of the total vacuum vessel height of 531 mm, as shown in Figure 5(b), the vertical displacement is determined to be 500 mm.

The horizontal side of the triangle, side a in Figure 25 corresponds to the electron's displacement in the toroidal direction. The value $2\pi R$ represents the path length of one complete turn along the toroidal magnetic field line at a distance R from the centerline. In our case, R is 85 mm. The total toroidal displacement depends on n, which represents the number of turns the electron completes along the toroidal field line.

The horizontal side (side a) and vertical side (side b) of the triangle in Figure 25 correspond to B_T and B_Z in Figure 24, respectively, as they represent the effects of B_T and B_Z on the electron's trajectory. Therefore, the ratio between the horizontal side and the vertical side equals to the ratio between B_T and B_Z. The smaller B_Z compared to B_T, the longer connection length L we can get.

Our goal is to ensure that B_Z is one-thousandth (1/1000) of B_T. Based on this ratio, if the vertical side length (corresponding to B_Z) in our calculation is 500 mm, then the horizontal side length (corresponding to B_T) would be 1000 times greater, which is 500 m.

For the length of the hypotenuse (side c), if θ in Figure 25 is sufficiently small (less than 0.1°), the hypotenuse can be approximated to the horizontal side. In our calculation, the vertical side of the triangle is one-thousandth (1/1000) of the horizontal side, so theta can be calculated as:

$$\theta = \arctan\left(\frac{\text{opposite}}{\text{adjacent}}\right) = \arctan\left(\frac{1}{1000}\right) = 0.057^\circ, \tag{56}$$

which satisfies the requirement. Therefore, the length of the hypotenuse can be considered the same as the length of the horizontal side, which is 500 m.

After calculating the connection length (L), we can determine the breakdown electric field (E_{BD}) using the Townsend coefficient (α), which describes the ionization rate of electrons as they travel through the gas. The physical meaning of α quantifies the number of ionization events caused by a single electron moving per unit length:

$$\alpha = A \cdot P_{\mathrm{He}} \cdot \exp\left(\frac{-B \cdot P_{\mathrm{He}}}{E}\right). \tag{57}$$

As mentioned in Chapter 2, the mini-Tokamak uses helium (He) as the working gas. For helium, the Townsend coefficient parameters are $A = 2.25\ 1/\mathrm{m} \cdot \mathrm{Pa}$ and $\mathrm{B} = 25.5\ \mathrm{V/m} \cdot \mathrm{Pa}$. The corresponding parameters for H_2, He, and Ar are shown in Table 1[14]. P_{He} is the gas pressure of He in Pa, and E is the electric field strength. To achieve plasma breakdown, the product of the Townsend coefficient (α) and the connection length (L) must satisfy:

$$\alpha \cdot L > 1\,, \tag{58}$$

indicates that the number of ionization events occurring along the connection length is sufficient to sustain a self-amplifying ionization process, leading to gas breakdown.

Table 1: Townsend coefficient parameters.

Gas	A $(1/\mathrm{m} \cdot \mathrm{Pa})$	B $(\mathrm{V/m} \cdot \mathrm{Pa})$
H_2	3.75	97.5
He	2.25	25.5
Ar	9.00	135.0

We combine Eq. (57) and Eq. (58) and apply some transformations to derive the minimum electric field required to achieve breakdown:

$$
\begin{aligned}
& A \cdot P_{\mathrm{He}} \cdot \exp\left(\frac{-B \cdot P_{\mathrm{He}}}{E}\right) \cdot L > 1 \\
\Rightarrow & \exp\left(\frac{-B \cdot P_{\mathrm{He}}}{E}\right) > \frac{1}{A \cdot P_{\mathrm{He}} \cdot L} \\
\Rightarrow & \exp\left(\frac{B \cdot P_{\mathrm{He}}}{E}\right) < A \cdot P_{\mathrm{He}} \cdot L \\
\Rightarrow & \frac{B \cdot P_{\mathrm{He}}}{E} < \ln\left(A P_{\mathrm{He}} L\right) \\
\Rightarrow & E > \frac{B \cdot P_{\mathrm{He}}}{\ln\left(A P_{\mathrm{He}} L\right)}
\end{aligned}
\tag{59}
$$

The last term in Eq. (59) represents the condition for achieving breakdown, indicating that the induced electric field in the vacuum must be greater than $(B \cdot P_{\mathrm{He}})/\ln(AP_{\mathrm{He}}L)$. Based on this condition, we can define the breakdown electric field (E_{BD}) as:

$$
E_{\mathrm{BD}} = \frac{B P_{\mathrm{He}}}{\ln(A P_{\mathrm{He}} L)} \tag{60}
$$

where E_{BD} is the breakdown electric field, P_{He} is the gas pressure of He in Pa, and L is the connection length.

Using the method described above, we can calculate the breakdown electric field under different conditions, such as background pressure and connection length, which depend on factors like the ratio between B_T and B_Z , the height of the vacuum vessel, and the distance from the centerline. Figure 26(a) shows the breakdown curve of the electric field for a gas pressure range from 10^{-3} Pa to 10^{0} Pa, with the ratio between B_T and B_Z varying from 0.1 to 0.001.

By calculating the breakdown curve of the electric field, we can further determine the breakdown voltage (V_{BD}), i.e., the required loop voltage for breakdown, at our desired plasma position:

$$
V_{\mathrm{BD}} = E_{\mathrm{BD}} \cdot 2\pi R \tag{61}
$$

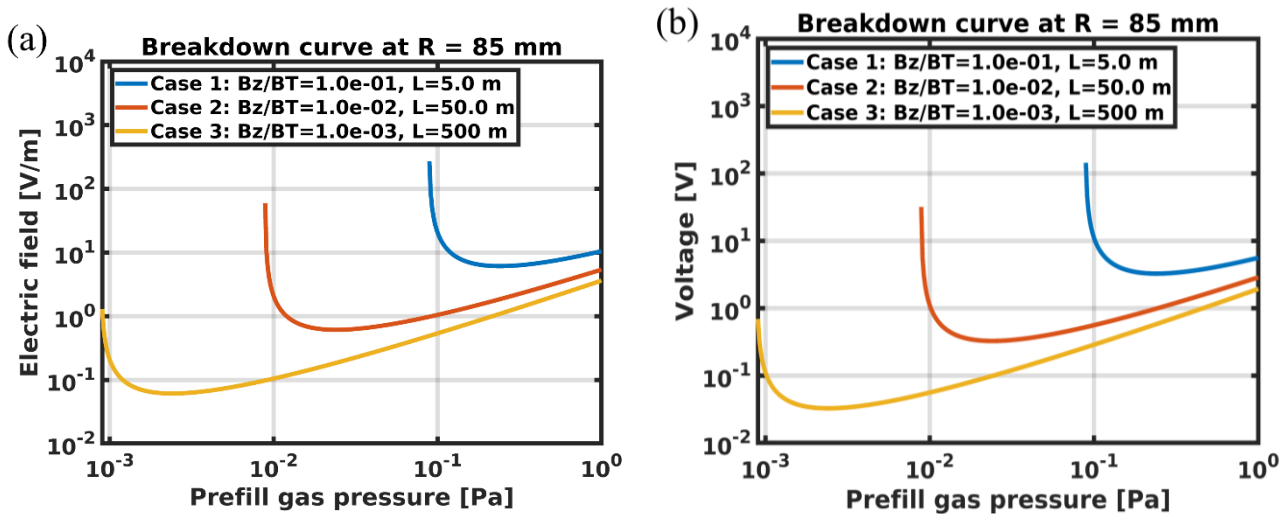

Figure 26: (a) Breakdown electric field curve. (b) Breakdown voltage curve.

where R is the distance from the centerline, which is 85 mm in our present calculation. Figure 26(b) illustrates the breakdown voltage curve corresponding to the breakdown electric field shown in Figure 26(a).

4.2. The required rate of change of central solenoid current

Using Eq. (60) and (61), we can calculate the breakdown voltage required to achieve under the target conditions of our experiment. For example, if the target condition is when B_Z is 0.1% of B_T, and the vacuum vessel height is 500 mm (at a radius of 85 mm), the connection length (L) is 500 m. Additionally, if the pressure is 10^{-4} Torr (equivalent to 1.3×10^{-2} Pa), i.e., the particle density is approximately 10^{18} m^{-3}, the breakdown electric field calculated using Eq. (60) is

$$E = \frac{BP}{\ln(APL)} = \frac{25.5 \times 1.3 \times 10^{-2}}{\ln(2.25 \times 1.3 \times 10^{-2} \times 500)} = 0.123\ {}^{\mathrm{V}}/_{\mathrm{m}}. \tag{62}$$

Based on this electric field, we can calculate the breakdown voltage required to generate plasma at a distance of 85 mm from the centerline using Eq. (61):

$$V = E \cdot 2\pi R = 0.123 \times 2\pi \times 0.085 = 0.066\ \mathrm{V}. \tag{63}$$

Therefore, we can calculate the required central solenoid current rate to induce this voltage in the vacuum vessel.

The calculation of the central solenoid current rate required to induce the target loop voltage is based on Faraday's law of electromagnetic induction. According to Faraday's law, the induced voltage (V) in a loop is proportional to the rate of change of magnetic flux through the loop:

$$V = -\frac{d\varphi}{dt} \tag{64}$$

where V is the induced voltage (loop voltage), and φ is the magnetic flux. The magnetic flux is defined as:

$$\varphi = \int BdA \approx B \times A \tag{65}$$

where B is the magnetic field strength, and A is the cross-sectional area of the loop.

For an ideal solenoid, the magnetic field B can be expressed in terms of the current I:

$$B = \mu_0 \frac{N}{l} I \tag{66}$$

where μ_0 is the permeability of free space ($4\pi \times 10^{-7}$ $^{\mathrm{H}}/_{\mathrm{m}}$), N is the number of turns in the coil, l is the length of the solenoid, and I is the current in the solenoid. Substituting B into Eq. (65):

$$\varphi = \mu_0 \frac{N}{l} I \times A. \tag{67}$$

By substituting Eq. (67) into Eq. (64):

$$V = -\frac{d}{dt}\left(\mu_0 \frac{N}{l} I \times A\right). \tag{68}$$

Since μ_0, N, l, and A are constants, Eq. (68) is simplified to:

$$V = -\mu_0 \frac{N}{l} A \frac{dI}{dt}. \tag{69}$$

Rearranging the equation, we get the rate of change of current:

$$\frac{dI}{dt} = -\frac{V \cdot l}{r^2 \pi \cdot N \cdot \mu_0}. \tag{70}$$

To calculate the rate of change of current required to induce the target loop voltage, we substitute the given values into the derived formula:

- $V = 0.066$ V: The target loop voltage that needs to be induced in the vacuum vessel, as calculated in Eq. (63).
- $l = 0.552$ m: The axial length of the solenoid over which the magnetic field is distributed, as described in Figure 6.
- $r = 0.011$ m: The major radius of the solenoid, which is used to calculate the cross-sectional area, was given in subsection 3.3.
- A: The cross-sectional area of the solenoid is calculated as:

$$A = \pi r^2 = \pi(0.011)^2 = 3.8 \times 10^{-4}\ \mathrm{m}^2. \tag{71}$$

- $N = 184$: The number of turns in the solenoid, contributing to the magnetic field strength.
- $\mu_0 = 4\pi \times 10^{-7}\ {}^{\mathrm{H}}/_{\mathrm{m}}$: The permeability of free space, a physical constant.

We get:

$$\frac{dI}{dt} = -\frac{0.066 \times 0.552}{\pi(0.011)^2 \cdot 184 \cdot 4\pi \times 10^{-7}} = -414.5\ {}^{\mathrm{A}}/_{\mathrm{ms}}. \tag{72}$$

The negative sign indicates that the current is decreasing over time, consistent with the direction of the induced voltage as described by Faraday's and Lenz's laws.

Through calculation, we determined that inducing a loop voltage of 0.066 V in the vacuum vessel and maintaining it for 15 ms requires a maximum central solenoid current of 6.22 kA. However, the 0.066 V breakdown voltage calculated in Eq. (63) represents the ideal conditions required for breakdown. In practice, various factors can influence the breakdown process. For example, if the actual B_Z is greater than one-thousandth of B_T, connection length would decrease, thereby increasing the required breakdown voltage. In this situation, generating only a 0.066 V loop voltage would not be sufficient to achieve breakdown and generate plasma.

As a result, the induced loop voltage is often several times higher than the calculated breakdown voltage. As shown in Table 2, if we aim to induce a loop voltage of 0.33 V, which is five times the breakdown voltage calculated in Eq. (63) to ensure reliable breakdown, the required maximum current of the central solenoid would be approximately 20 kA. Alternatively, we can reduce the time duration (t_{duratin}) while maintaining the same current rate ($I_{\max}/t_{\mathrm{duratin}}$) to lower the required maximum current of the central solenoid.

Since the maximum central solenoid current in our system is limited to 10 kA, we prefer either using the case with an induced loop voltage of 0.1 V for 15 ms, or shortening the duration to 10 ms with the full 10 kA current, which yields a loop voltage of approximately 0.17 V. This value is about 2.5 times the calculated breakdown voltage of 0.066 V, which should be sufficient to initiate breakdown. However, the detailed solenoid current profile will be shown in Chapter 5, considering plasma parameters and their time evolution.

Table 2: Central solenoid current requirements for different loop voltages and durations.

V_{loop}	t_{duratin}	$I_{\max}$	dI/dt
0.066 V	15 ms	6.22 kA	415 A/ms
0.1 V	15 ms	9.4 kA	627 A/ms
0.33 V	10 ms	20.7 kA	2070 A/ms

4.3. Calculation of the eddy currents

A simple model given in Section 4.2 provided a guideline for determining the loop voltage and thus the required current rate of the central solenoid. However, eddy currents on the vacuum-vessel walls can also influence the loop voltage. They also need to be considered carefully. Therefore, we developed a program to calculate the breakdown voltage under specific conditions. We use $dI/dt = -415\ \mathrm{A/ms}$ as an example to determine the eddy currents calculation. Assuming, it takes 15 ms for current to drop from 6.2 kA to 0. Additionally, the program incorporates calculations from Chapter 3 to determine the eddy currents induced in the inner and outer vacuum-vessel walls by the

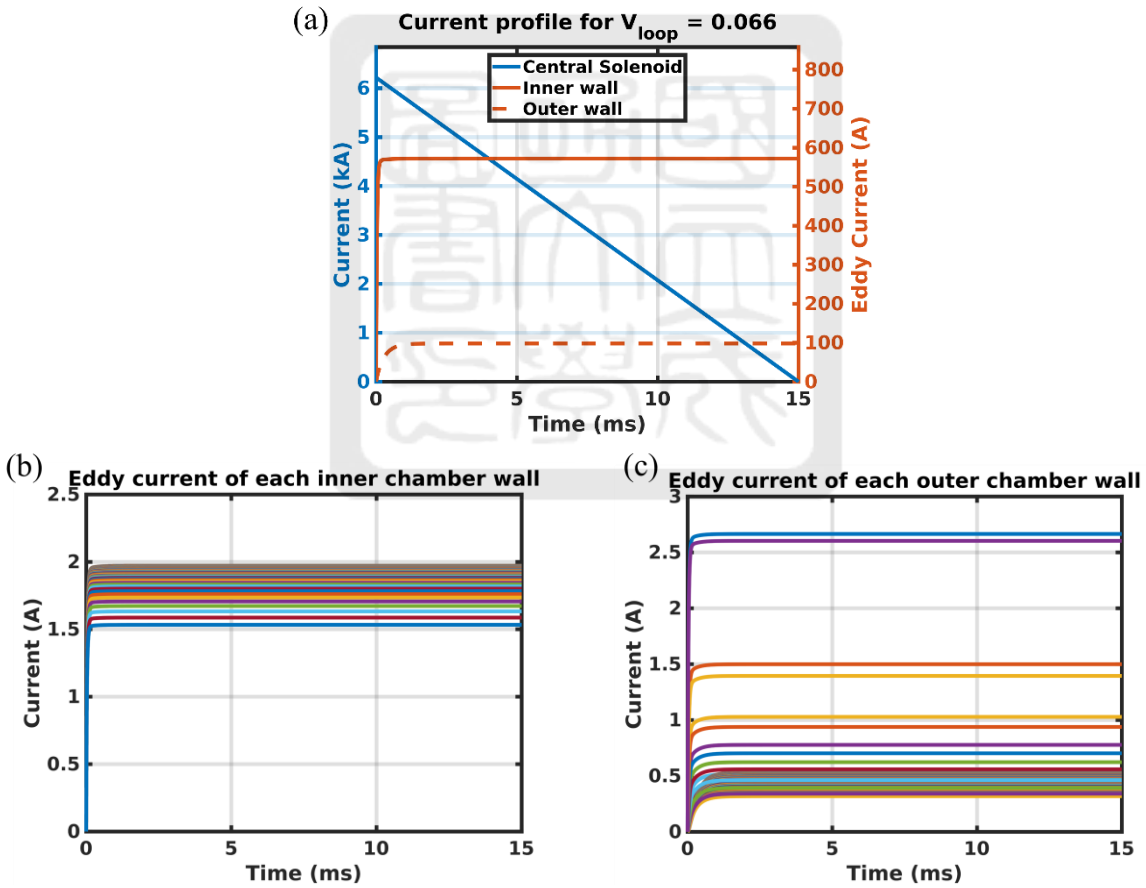

Figure 27: (a) Current profile of central solenoid and the eddy currents of the inner and outer vacuum-vessel wall. (b) The eddy current of each element of the inner-vacuum-vessel wall. (c) The eddy current of each element of the outer-vacuum-vessel wall.

time-varying current of the central solenoid, based on the current profile. Finally, the induced voltage can be calculated from all currents including the current of the central solenoid and eddy currents. It will be introduced in Section 4.4. As shown in Figure 27(a), the eddy current in the inner vacuum-vessel walls corresponds to the red solid line, while the eddy current in the outer vacuum-vessel walls corresponds to the red dashed line. Notice that the program first calculates the eddy current in each inner and outer vacuum-vessel walls individually as shown in Figure 27(b) and (c), respectively. The currents displayed in the Figure 27(a) represent the total current, obtained by summing all the contributions from the inner walls and outer walls, respectively.

4.4. Calculation of the exact induced loop voltage

Using the program described in Section 4.3, we had the current profile of the central solenoid and the individual element of the inner-and-outer-vacuum-vessel walls. With the current profile, we can determine the magnetic field distribution produced by each element and further calculate the total magnetic flux. By using Eq. (64), the loop voltage is calculated from the rate of change of total magnetic flux. Notice that we only calculate the loop voltage in the equatorial plane. Therefore, by back-calculating from the magnetic flux variations, we can verify whether the current profile generated by the program in Section 4.3 can induce the required loop voltage in the vacuum vessel.

First, we use the Biot-Savart Law to calculate the magnetic field distribution from each element, which represents the vacuum-vessel-wall elements. It is important to note that since the tokamak is axisymmetric, the elements are also axisymmetric. Consequently, the magnetic field distribution only needs to be calculated in the xz-plane along the positive x-direction, as illustrated in Figure 28.

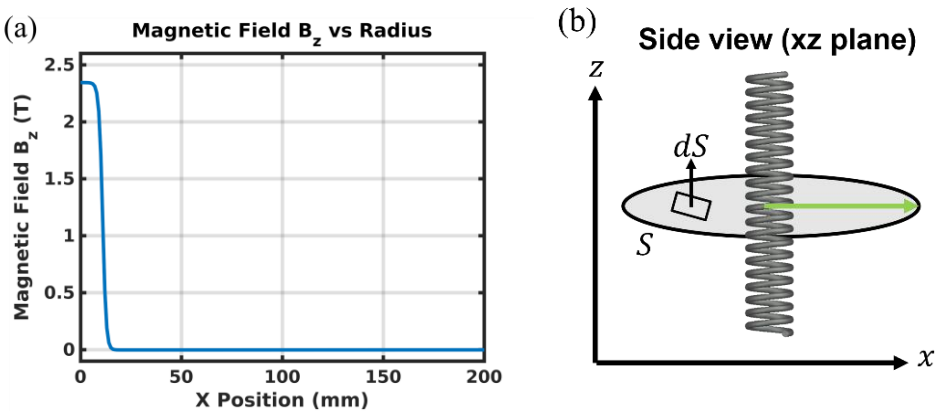

Figure 28: (a) Magnetic field B_Z in the x-direction at z = 0 and y = 0. (b) Schematic of the central solenoid.

Similarly, we use the Biot-Savart Law to determine its magnetic field distribution from the central solenoid. Due to its axisymmetric, the calculation can be restricted to a line on the equatorial plane (i.e., $z = 0$). In this line, we only calculate the magnetic field component B_Z at different location in x, corresponding to the green line shown in Figure 28(b).

The equatorial plane (i.e., $z = 0$), represented by the gray circular plane in Figure 28(b), are chosen because the tokamak's axisymmetric structure ensures the magnetic field is symmetric about the central axis, i.e., $\partial\vec{B}/\partial\phi = 0$. The magnetic field, originally a function of x and y, can therefore be simplified to a function of the radial coordinate $r = \sqrt{x^2 + y^2}$, i.e., $B(x, y) = B(r)$. This allows the magnetic flux calculation to be reduced to a one-dimensional integral:

$$\int B \cdot dA = \int B \cdot 2\pi r \cdot dr = 2\pi \int B \cdot r \cdot dr\,. \tag{73}$$

Furthermore, only B_Z is calculated because it is the only component of the magnetic field that is perpendicular to the loop surface and determines the loop voltage in the tokamak.

After calculating the time-dependent changes in the magnetic field distribution, the magnetic flux can be determined by integrating the magnetic field along the green line, as shown in Figure 28(b). The loop voltage is then calculated:

$$V = -\frac{d\varphi}{dt}. \tag{74}$$

Using the above method, we calculated the loop voltage induced solely by the central solenoid current rate, as shown in Figure 29. For the case where the current profile dropped from 6.2 kA to 0 in 15 ms, as shown by the blue line in Figure 27(a), which the resulting loop voltage calculated by the model is 0.063 V. It is worth noting that the central solenoid current profile used for this calculation, as determined in Subsection 4.2, was designed to induce 0.066 V. The reason the induced voltage only 0.063 V is because the calculations in Subsection 4.2 were based on ideal solenoid equations. However, in practical scenarios, various factors can reduce the induced loop voltage. Therefore, it is necessary to perform back-calculations to benchmark the accuracy and reliability of our calculation and to evaluate the performance of the central solenoid design.

Next, we applied the same approach to calculate the magnetic field distribution produced by each element of the inner-and-outer-vacuum-vessel-walls along a line of $y =$

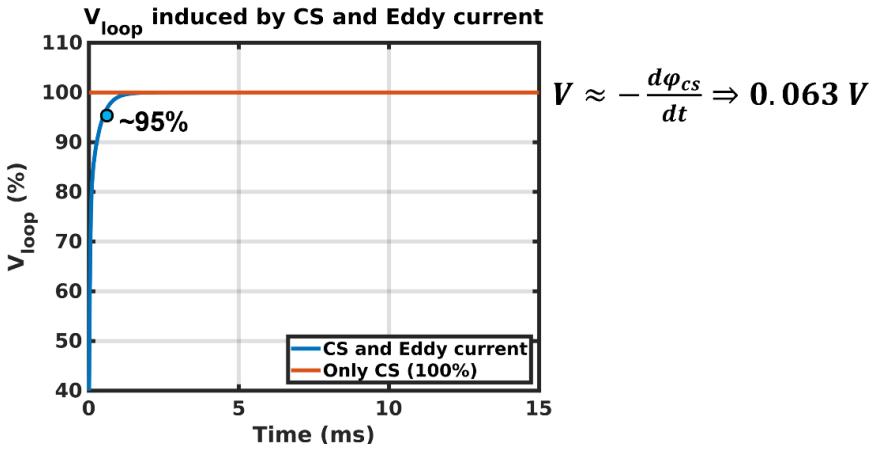

Figure 29: Comparison of loop voltage induced by central solenoid and eddy currents in the vacuum-vessel walls over time.

$z = 0$. After summing fields generated from all components, we integrated the total magnetic field to calculate the magnetic flux. Finally, using Eq. (64), we obtained the loop voltage induced under the influence of eddy currents in the vacuum-vessel walls, as represented by the blue line in Figure 29.

From Figure 29, it can be observed that around 0.5 ms, the influence of eddy currents in the vacuum-vessel walls on the loop voltage reduces to approximately 5%. This is a key point as it indicates that a central solenoid current profile with a duration longer than 0.5 ms, the impact of eddy currents in the vacuum-vessel walls on the loop voltage can be ignored.

The detailed code is provided in Appendix A.13, including the calculation of the required current ramp rate to induce the expected loop voltage, the calculation of eddy currents induced in the vacuum-vessel walls, and the exact induced loop voltage.

4.5. Discussion

In our current calculation of the breakdown voltage, only direct electron-impact ionization is considered. However, in practice, the actual breakdown voltage can be very different from that considered electron-impact ionization alone. This difference arises because ionization is not solely driven by direct electron collisions; excitation processes and the formation of metastable states also play important roles.

Moreover, helium gas is inherently difficult to ionize through electron collisions alone. This is due to helium's high first ionization energy (approximately 24.6 eV), which requires electrons to gain sufficient energy and undergo multiple collisions before effective ionization can occur. As a result, plasma breakdown in a pure helium environment is often difficult to achieve with electron impact alone.

To address this limitation, we may consider utilizing the Penning effect in future experiments to assist the breakdown process. This can be achieved by introducing a small amount of gas with a lower ionization energy, such as neon or argon. When helium atoms

are excited to metastable states, collisions with these additives can release enough energy to ionize them, thereby facilitating the breakdown and effectively reducing the required voltage.

In addition, our current calculation assumes B_Z is one-thousandth (1/1000) of B_T. That is, for a toroidal field strength B_T = 0.1 T, the vertical component B_Z is set to only 10^{-4} T. This configuration is intended to produce a magnetic null field at R = 85mm, where plasma breakdown is expected to occur. However, as shown in Figure 23, the central solenoid inevitably produces a stray vertical magnetic field in the upward direction. To cancel this field and achieve the desired null point, the poloidal field coils (PFCs) must generate a downward B_Z. However, this magnetic field is different from the equilibrium magnetic field configuration. Therefore, once the plasma is formed, the PFCs would need to rapidly reverse their current direction in order to maintain a stable magnetic equilibrium. This rapid current reversal is technically very difficult to achieve in practice. Consequently, maintaining such low B_Z and dynamically switching the coil polarity poses significant engineering challenges, and may limit the practical implementation of the idealized null-field scenario used in the model. As a result, the actual ratio (B_Z/B_T) may be closer to 1/100 or even 1/10. This significantly reduces the connection length of particles from the estimated 500 m to roughly 50 m or even 5 m, which in turn increases the required breakdown voltage by more than an order of magnitude.

Nevertheless, our current calculation is based on these simplified assumptions, with the primary goal of obtaining a rough estimate of the required breakdown voltage to support hardware design and planning. Therefore, we continue to adopt the model calculation results at this stage. The actual required voltage will be confirmed in the future through more detailed simulations or experimental validation.

4.6. Conclusion

In this work, we calculated the required breakdown voltage for the mini-Tokamak and determined the central solenoid current profile necessary to induce the corresponding loop voltage. The induced loop voltage was further computed using a model that incorporates the effects of eddy currents in the vacuum-vessel-walls, in conjunction with the model developed in Chapter 3. Our results show that the impact of eddy currents on the loop voltage is negligible under the mini-Tokamak's conditions. This method allows us to benchmark the accuracy and reliability of our calculations and to evaluate the performance of the central solenoid design. It also enables further optimization of the design to meet experimental requirements.

5. Calculation of the evolution of plasma parameters

In the previous section, we calculated the breakdown voltage required to generate plasma, the central solenoid current profile to induce the required loop voltage in the vacuum for breakdown and calculate the induced loop voltage including the effect of eddy currents on the vacuum-vessel walls. Based on these calculations, we assume that breakdown has happened so that plasma has been generated. Then the loop voltage can drive the plasma current generating the poloidal magnetic field. In addition, plasma is heated by the plasma current through Ohmic heating. To model the plasma parameters, plasma can be modeled as a single-turn elliptical coil inside the vacuum vessel as shown in Figure 30. The elliptical cross-section of the coil is centered on the z = 0 plane, 85 mm from the centerline. The minor axis of the coil is 55 mm. The elongation factor κ = 1.82, which represents the ratio of the major axis of the ellipse to its minor axis, gives the coil with the major axis of 100.1 mm.

As mentioned in Section 3.1, the full circuit equation, which accounts for mutual inductance between different components, allows us to calculate the induced currents between various components within the tokamak device. Similarly, the plasma is modeled as a single-turn coil, represented by the blue torus shown in Figure 30. The induced current in the plasma, primarily generated by the central solenoid, is also influenced by eddy currents

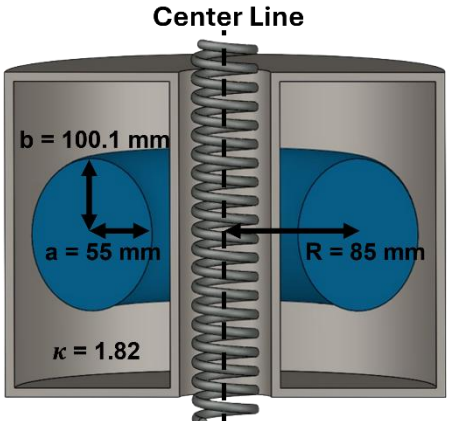

Figure 30: Simplified xz-plane cross-section of mini-Tokamak.

in the vacuum-vessel walls. By determining the plasma's resistance, self-inductance, and mutual inductance with other components, we can calculate the induced current within the plasma, which is referred to as the plasma current.

Once the plasma current is known, we can calculate the energy deposited into the plasma through Ohmic heating. Plasma parameters such as temperature, ionization fraction, and plasma density can then be calculated. The plasma resistivity can be updated in every time step as the feedback to the circuit model. Using this, we can determine the evolution of plasma parameters over time. This analysis allows us to evaluate whether the designed central solenoid current profile can achieve our experimental goals, which include a plasma temperature exceeding 10 eV and a plasma current greater than 10 kA. With the model, we can design and adjust the central solenoid current profile to ensure the plasma parameters meet our targets.

This chapter consists of six sections. Section 5.1 explains the plasma parameters that need to be calculated, including plasma resistivity, temperature, and density. Section 5.2 presents the calculation of the plasma current, and the resulting design of the central solenoid current profile based on this model. Section 5.3 demonstrates the sensitivity test of the model with respect to the initial temperature, in order to verify the robustness and physical consistency of the model. Section 5.4 introduces neoclassical theory, which accounts for trapped particles and tokamak geometry effects that are not considered in the Spitzer theory. Section 5.5 presents the designed central solenoid current profile and the resulting calculated plasma parameters. Section 5.6 discusses the limitations and assumptions, and Section 5.7 provides the conclusion.

5.1. Calculation of plasma parameters

To calculate the plasma current, we first need to determine the plasma resistance and inductance. For simplicity, the spatial profile and the location of the plasma are kept fixed. In other words, plasma geometry shown in Figure 30 is kept fixed in this work. The self-inductance of the plasma L_{p} can be calculated using the following equation[15]:

$$L_{\mathrm{p}} = \mu_0 R \left(\ln \frac{8R}{a} + \frac{l_{\mathrm{i}}}{2} - 2 \right) \tag{75}$$

where $\mu_0 = 4\pi \times 10^{-7}\ \mathrm{H/m}$ is the permeability of free space, R is the distance from the centerline, which is 85 mm in our calculation and corresponds to R in Figure 30. The parameter a represents the minor radius of the plasma, which is 55 mm and corresponds to a in Figure 30. The parameter l_{i}, which represents the internal inductance of the plasma, is a function of the current distribution in the cross-section. For simplicity in our calculation, we assume $l_i = 0.5$.

As for the mutual inductance, since Eq. (49) and Eq. (54) depend only on the position of the coil filaments and is independent of the coil's minor radius, we directly use them to determine the mutual inductance between the plasma and tokamak components, as the plasma is coaxial with them in the ideal case. Finally, we obtained the necessary inductance for matrix $\mathbf{M}$ in the full circuit equation.

For matrix $\mathbf{R}$, since plasma parameters that affect plasma resistance change over time such as temperature and density. Before calculating the plasma current, we must first determine how these plasma parameters evolve over time.

The plasma resistivity can be calculated using the Spitzer equation[15]:

$$\eta_{\mathrm{Sp}} = 5.2 \times 10^{-3} \cdot Z \cdot \ln\Lambda \cdot T_{\mathrm{e}}^{-1.5} \cdot 10^{-2} \tag{76}$$

where η_{Sp} is the Spitzer resistivity, Z is the charge state of the ions; T_{e} is the electron temperature in electron volts (eV), and $\ln\Lambda$ is the Coulomb logarithm[16], which can be determined by:

$$\ln\Lambda = 23.5 - \log\left(\frac{\sqrt{n_e \times 10^{-6}}}{T_e^{\frac{5}{4}}}\right) - \sqrt{10^{-5} + \frac{(log(T_e) - 2)^2}{16}}. \quad (77)$$

In our present experiment, we use helium (He), which can exist in charge states $Z = 1$ and $Z = 2$. However, the plasma temperature in the mini-Tokamak does not exceed 10 eV. Therefore, for simplicity, we assume $Z = 1$ in our calculation.

The plasma resistance can be calculated using the Spitzer resistivity given in Eq. (76) as follows:

$$R_p = \eta_{Sp}\frac{L}{A} = \eta_{Sp}\frac{2\pi R}{\pi ab} \quad (78)$$

where R_p is the plasma resistance, L is the effective length of the plasma current path, which is the length of the plasma coil of the blue toroidal single-coil shown in Figure 30 given by $2\pi R$, $R = 85$ mm is the major radius of plasma. The parameters $a = 55$ mm and $b = 100.1$ mm represent the minor and major axes of the plasma cross-section, respectively. The cross-sectional area of the plasma is given by $A = \pi ab$.

It can be observed that η_{Sp} and $\ln\Lambda$ are affected by plasma density and temperature, both of them vary over time due to ionization and Ohmic heating. Therefore, before calculating plasma resistance, we must first determine how ionization fraction and plasma temperature evolve over time.

5.1.1. Calculation of plasma density

Plasma density is determined by the balance between the ionization rate and the recombination rate. In our model, we use data from the Atomic Data and Analysis Structure (ADAS)[17] for high temperature and ionization fraction to calculate both the ionization and recombination rates. ADAS is a comprehensive database and toolset that provides accurate atomic and ionic data for plasma modeling. It is widely used in fusion energy research, astrophysics, and laboratory plasma studies to support calculations of key processes such as ionization, recombination, and radiation emission.

The electron temperature range covered by ADAS spans from 0.2 eV to 1×10^4 eV, while the electron density ranges from 5×10^{13} m^{-3} to 2×10^{21} m^{-3}. In our current experiment, the plasma is expected to reach temperature around 10 eV and particle density between 10^{17} m^{-3} to 10^{18} m^{-3}. Since these values fall well within the range covered by the ADAS datasets, we are able to apply these data to our model without the need for extrapolation.

However, ADAS does not provide reliable data for low-temperature and low-density conditions. This limitation arises because the ionization potentials of hydrogen and helium are 13.6 eV and 24.6 eV, respectively. At low electron energies, excitation and ionization events are rare, and the corresponding cross sections become very small, making the coefficients difficult to measure or compute accurately. Therefore, for low-density and low-ionization-fraction conditions, we adopt a simplified collisional model:

$$n_e \sigma_{e-i} = (n_a - n_e)\sigma_{e-a}\,. \tag{79}$$

Assuming a balance between ionization and neutralization, this equation defines the relationship between electron density and neutral atom density during the early phase of plasma formation. Details are given in the following.

5.1.1.1. ADAS model

There are several methods for calculating the ionization fraction. However, to save time and improve efficiency, we use precomputed rate coefficients derived from established models and theories in ADAS[17] for high temperature and high ionization fraction.

We can calculate the plasma density using the following equation:

$$\frac{dn_e}{dt} = R_{ion}(T_e, n_e) n_0 n_e - R_{rec}(T_e, n_e) n_e n_i. \tag{80}$$

The variable n_e represents the electron density, which is considered equivalent to the plasma density. The ion density, denoted as n_i, is assumed to be equal to the electron density

($n_\mathrm{i} = n_\mathrm{e}$) since ionization releases both an electron and an ion in our case. The neutral atom density (n_0) is determined based on our calculation in Subsection 4.2, where the experimental pressure is 10^{-4} Torr (equivalent to 1.3×10^{-2} Pa). Given these conditions, the initial particle density is approximately $10^{18}\mathrm{m}^{-3}$, so we assume an initial neutral atom density of $n_0 = 10^{18}\ \mathrm{m}^{-3}$. $R_\mathrm{ion}(T_\mathrm{e}, n_\mathrm{e})$ is the effective ionization coefficient, and $R_\mathrm{rec}(T_\mathrm{e}, n_\mathrm{e})$ is the effective recombination coefficient; both can be obtained from the ADAS database[17]. These correspond to the SCD (effective ionization coefficients) and ACD (effective recombination coefficients) datasets, respectively.

Eq. (80) represents the rate of change of electron density by accounting for both ionization and recombination processes. The first term describes electron generation due to ionization, while the second term accounts for electron loss due to recombination.

5.1.1.2. Collision model

For low temperature and low ionization fraction case, we use collision model to calculate the ionization fraction. The initial ionization fraction can be calculated using the ratio of electron-atom to electron-ion collision cross-sections[15]:

$$n_\mathrm{e}\sigma_\mathrm{e-i} = (n_0 - n_\mathrm{e})\sigma_\mathrm{e-a}, \tag{81}$$

where n_e is the electron density in m^{-3}, n_0 is the total particle density in m^{-3}, which is related to the working gas pressure of the experiment and is assumed constant. σ_{e-i} and $\sigma_\mathrm{e-a}$ are the electron–ion and electron–atom collision cross-sections in m^2, respectively. T_e is the electron temperature in eV.

The left-hand side of Eq. (82) represents the rate of electron-ion collisions, which is determined by the product of the electron density (n_e) and the electron-ion collision cross-section ($\sigma_\mathrm{e-i}$). It means the neutralization rate. The right-hand side represents the rate of electron-atom collisions, which depends on the neutral atom density $(n_0 - n_\mathrm{e})$ and the electron-atom collision cross-section ($\sigma_\mathrm{e-a}$). It means the ionization rate. Assuming the

ionization rate and the neutralization rate are balanced with each other, this equation defines the relationship between electron density and neutral atom density during the early plasma formation phase. By rearranging Eq. (82), we obtain:

$$\frac{n_{\mathrm{e}}}{n_0 - n_{\mathrm{e}}} \approx \frac{\sigma_{\mathrm{e-a}}}{\sigma_{\mathrm{e-i}}} . \tag{82}$$

At the early stage of plasma ionization, the neutral gas density $(n_0 - n_{\mathrm{e}})$ is much higher than the electron density n_{e} because the ionization process has just begun. Since only a small fraction of atoms has been ionized, it is reasonable to assume $n_0 \gg n_{\mathrm{e}}$. Therefore, when calculating $n_0 - n_e$, we can approximate it as n_0 by neglecting n_e. This allows us to express the ionization fraction as the ratio between n_{e} and n_0:

$$\gamma \approx \frac{n_{\mathrm{e}}}{n_0} \approx \frac{\sigma_{\mathrm{e-a}}}{\sigma_{\mathrm{e-i}}}. \tag{83}$$

Therefore, once the cross-sections $\sigma_{\mathrm{e-a}}$ and $\sigma_{\mathrm{e-i}}$ are expressed as functions of electron temperature, the ionization fraction γ can be directly calculated as a function of temperature.

(a) Electron-atom collision cross-section

For hydrogen, according to an empirical formula, $\sigma^{\mathrm{H}}_{\mathrm{e-a,formula}}(T_{\mathrm{e}}) \approx 3 \times 10^{-19} \times T_e^{-0.5}$[15]. However, since the mini-Tokamak uses helium as the working gas, we need an equivalent expression for $\sigma^{\mathrm{He}}_{\mathrm{e-a,formula}}(T_{\mathrm{e}})$.

Direct empirical formulas for helium are less commonly available, but we can infer one by analyzing the ratio of ionization cross-sections between hydrogen and helium over the relevant temperature range. Specifically, by evaluating or interpolating $\sigma^{\mathrm{H}}_{\mathrm{e-a,ADAS}}(T_{\mathrm{e}})$ and $\sigma^{\mathrm{He}}_{\mathrm{e-a,ADAS}}(T_{\mathrm{e}})$ from datasets such as ADAS, we can compute the ratio $\sigma^{\mathrm{He}}_{\mathrm{e-a,ADAS}}(T_{\mathrm{e}})/\sigma^{\mathrm{H}}_{\mathrm{e-a,ADAS}}(T_{\mathrm{e}})$ across a range of electron temperatures.

In our study, we use the Effective Ionization Coefficients (SCD) for both hydrogen and helium. These rate coefficients, provided in units of cm^3/s in the raw

ADAS data, represent the effective ionization reaction rates per unit electron and target particle density, which are functions of both the electron temperature and the electron density. To convert these rate coefficients into effective cross sections (in m²), we use the relation:

$$\sigma_{\mathrm{e-a}} = \frac{R_{\mathrm{ion}} \times 10^{-6}}{v_{\mathrm{e}}}, \tag{84}$$

where R_{ion} is the rate coefficient in cm³/s, and $v_{\mathrm{e}} = \sqrt{2eT_{\mathrm{e}}/m_{\mathrm{e}}}$ is the electron thermal velocity in m/s. After interpolating the ADAS tables, we compute the ionization cross sections for both hydrogen and helium, as shown in Figure 31(a).

Assuming this ratio captures the relative magnitude and trend of helium behavior compared to hydrogen, we can scale the hydrogen formula by the ratio to obtain an approximate expression for helium, as shown in Figure 31(b):

$$\sigma^{\mathrm{He}}_{\mathrm{e-a,formula}}(T_{\mathrm{e}}) \approx \frac{\sigma^{\mathrm{He}}_{\mathrm{e-a,ADAS}}(T_{\mathrm{e}})}{\sigma^{\mathrm{H}}_{\mathrm{e-a,ADAS}}(T_{\mathrm{e}})} \cdot \sigma^{\mathrm{H}}_{\mathrm{e-a,formula}}(T_{\mathrm{e}}). \tag{85}$$

In Figure 31(b), the blue line is the empirical formula of the ionization cross section of hydrogen from Ref. [15] while the red line is the estimation of that of helium using Eq. (85). This approach preserves the functional dependence of the hydrogen

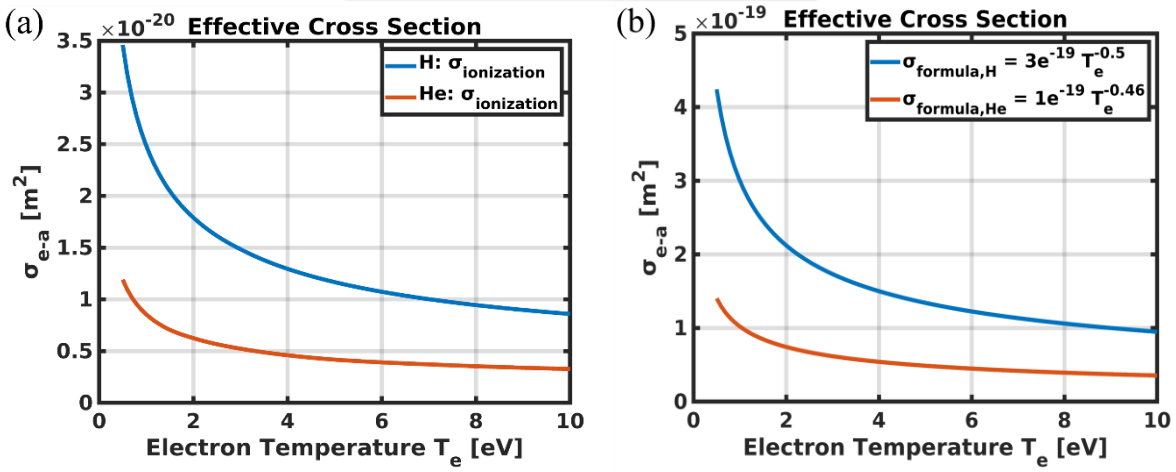

Figure 31: (a) ADAS-based effective cross sections. (b) Empirical formula for hydrogen and fitted formula for helium.

formula while adjusting the amplitude and temperature exponent to better match helium data. We can then fit the scaled expression to a power-law form:

$$\sigma_{\mathrm{e-a,formula}}^{\mathrm{He}}(T_{\mathrm{e}}) \approx A \times T_e^{-b} \approx 1 \times 10^{-19} \times T_e^{-0.46}, \tag{86}$$

where A and b are determined through curve fitting.

(b) Electron-ion collision cross-section

We adopt an empirical approach to estimate the electron–ion collision cross section under low-temperature conditions relevant to our mini-Tokamak system. We use the empirical formula for hydrogen[15]:

$$\sigma_{\mathrm{e-i}}^{\mathrm{H}}(T_{\mathrm{e}}) \approx 1.5 \times 10^{-16} \times T_{\mathrm{e}}^{-2}, \tag{87}$$

Although this formula is derived specifically for hydrogen, we apply it to helium as well by assuming that all helium atoms are singly ionized (Z = 1), which is valid in our case since the plasma temperature remains around 10 eV.

Based on the electron-atom collision cross-sections presented in Figure 31(b), where the blue solid line corresponds to hydrogen and the red solid line to helium, and the empirical formula for electron-ion collision cross-sections. The ionization fraction can be calculated using Eq. (83). For hydrogen:

$$\gamma_{\mathrm{H}} \approx \frac{3 \times 10^{-19} \times T_{\mathrm{e}}^{-0.5}}{1.5 \times 10^{-16} \times T_{\mathrm{e}}^{-2}} = 2 \times 10^{-3} \times T_{\mathrm{e}}^{1.5}\ \mathrm{eV}. \tag{88}$$

For helium:

$$\gamma_{\mathrm{He}} \approx \frac{1 \times 10^{-19} \times T_{\mathrm{e}}^{-0.46}}{1.5 \times 10^{-16} \times T_{\mathrm{e}}^{-2}} = 6.7 \times 10^{-4} \times T_{\mathrm{e}}^{1.54}\ \mathrm{eV}. \tag{89}$$

5.1.1.3. Transition from collision model to ADAS model

During the initial phase of plasma breakdown, we apply Eq. (88) and Eq. (89) for hydrogen and helium, respectively. These are referred to as the collision model and are used to calculate the initial ionization fraction. As the plasma evolves and becomes more ionized, we gradually transition to using Eq. (80), known as the ADAS model. This transition is

implemented using a weighted interpolation method, ensuring a smooth shift between the two models as a function of electron temperature or ionization fraction.

When $\gamma = 0$, where γ is the ionization fraction, the plasma is in the early stage of breakdown, and the ionization is calculated using the collision model. As the ionization fraction increases, the contribution from the ADAS model gradually increases. When $\gamma = 0.5$, the ADAS model dominates the ionization calculation. The weighting is controlled by a sigmoid function defined over the interval [0, 0.5]:

$$w_{\mathrm{ADAS}}(\gamma) = \frac{1}{1 + \exp[-s \cdot ((\gamma - \gamma_0)/\Delta]}. \tag{90}$$

where $s = 10$ controls the steepness of the transition. A larger value of s leads to a sharper transition between the two models, $\gamma_0 = 0.25$ is the center of the transition, and $\Delta = 0.5$ is the normalization range. The collision model weight is defined as $w_{\mathrm{collision}} = 1 - w_{\mathrm{ADAS}}$, ensuring the total contribution remains unity at all times.

At $\gamma = 0$, the ADAS model contributes approximately 0.67%, while the collision model dominates with over 99% weight. Around $\gamma = 0.25$, both models contribute equally, and by $\gamma = 0.5$, the ADAS model reaches nearly full weight (~99.3%) while the collision model becomes negligible (0.67%). This transition is illustrated in Figure 32, which shows the evolution of model weights as a function of γ.

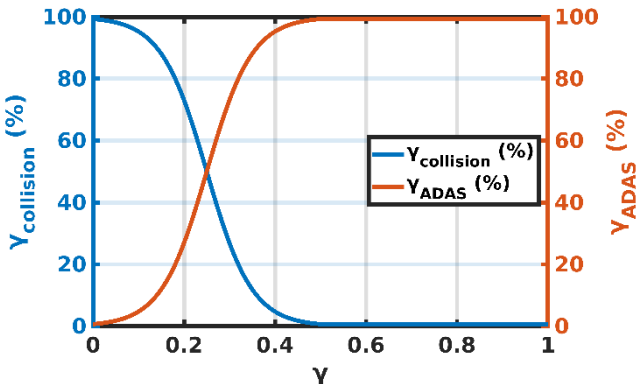

Figure 32: Weight distribution between collision model and ADAS model.

This approach allows for a smooth and physically consistent between two methods, avoiding abrupt changes in source terms and improving the accuracy of time-dependent plasma ionization modeling. It also ensures that the numerical system remains well-behaved during rapid changes in plasma parameters, which is particularly important in early-stage breakdown or startup simulations.

By implementing this approach, we can calculate the time-dependent evolution of plasma density, maintaining physical consistency across different phases of ionization.

5.1.2. Calculation of plasma temperature

The change in electron energy (thermal energy) is determined by the balance between heating and power losses. Therefore, to calculate the time-dependent variation of plasma temperature, we use the following equation[15]:

$$\frac{3}{2}\frac{d(n_e T_e)}{dt} = P_{oh} - \left(P_{ion} + P_{line} + P_{prb}\right) \tag{91}$$

where each term on the right-hand side represents a volumetric power density (W/m³).

The first term, P_{oh}, is the Ohmic heating power density and is considered the main energy source in the simulation. It is calculated as:

$$P_{oh} = \frac{I_p^2 R_p}{V_p} \tag{92}$$

where I_p is plasma current (A), R_p is plasma resistance (Ω) given in Eq. (78), and V_p is plasma volume (m³).

The ionization power loss P_{ion} accounts for the energy required to ionize neutral atoms, thereby diverting energy away from electron heating. It is computed by:

$$P_{ion} = R_{ion} \times n_e \times n_0 \times E_i \tag{93}$$

where R_{ion} is the effective ionization coefficient, obtained from ADAS[17] in units of m^3/s, n_e is the electron density in units of m^3, n_0 is the neutral atom density in units of m^3, and E_i is the ionization energy in J. The expression reflects that each ionization event

requires a fixed amount of energy, and the ionization rate depends on the product of electron and neutral densities.

The line radiation loss P_{line} arises from bound–bound transitions when excited ions return to lower energy states, emitting photons. It is given by:

$$P_{\mathrm{line}} = R_{\mathrm{plt}} \times n_{\mathrm{e}} \times n_{\mathrm{i}} \tag{94}$$

where R_{plt} is the coefficient of line emission from excitation in units of $\mathrm{W} \cdot \mathrm{m}^3$, obtained from ADAS[17], and $n_{\mathrm{i}} \approx n_{\mathrm{e}}$ under the quasi-neutrality assumption. This equation is based on the fact that line radiation occurs when electrons excite ions to higher energy levels, and these ions then emit photons upon returning to lower energy states. The excitation rate depends on electron-ion interactions, making the emission power proportional to the electron and ion densities.

In addition to line radiation, the simulation also includes Bremsstrahlung radiation loss, denoted R_{prb}. Bremsstrahlung, or free–free radiation, is caused by the deceleration of electrons in the electric field of ions, resulting in photon emission. The power density is computed using:

$$P_{\mathrm{prb}} = R_{\mathrm{prb}} \times n_{\mathrm{e}} \times n_{\mathrm{i}} \tag{95}$$

where R_{prb} is the Bremsstrahlung emission coefficient, interpolated from ADAS[17] in units of $\mathrm{W} \cdot \mathrm{m}^3$. This term is generally several orders of magnitude smaller than ionization and line radiation losses under low-Z, sub-keV conditions, and can often be neglected. Nevertheless, it is still tracked here for completeness.

By incorporating all these effects, the energy balance equation provides a consistent framework for simulating the plasma's thermal behavior. It ensures that the rise in electron temperature is not only driven by Ohmic heating but also realistically modulated by competing loss processes including ionization, line radiation, and Bremsstrahlung.

5.1.3. Government equations for calculating plasma parameters

In summary, the government equations for calculating plasma parameters are the following. Notice that particle losses and the corresponding energy losses are neglected in this simple model.

Full circuit equation for the induced current:

$$\vec{V} = \mathbf{M}\frac{d\vec{I}}{dt} + \mathbf{R}\vec{I}\,. \tag{96}$$

Equation for plasma inductance:

$$L_{\mathrm{p}} = \mu_0 R\left(\ln\frac{8R}{a} + \frac{l_{\mathrm{i}}}{2} - 2\right) \tag{97}$$

Equation for plasma resistivity and resistance:

$$\eta_{\mathrm{Sp}} = 5.2\times10^{-3}\cdot Z\cdot\ln\Lambda\cdot T_{\mathrm{e}}^{-1.5}\cdot10^{-2}, \qquad R_{\mathrm{p}} = \eta_{\mathrm{Sp}}\frac{L}{A} = \eta_{\mathrm{Sp}}\frac{2\pi R}{\pi ab} \tag{98}$$

Energy balance equation for the plasma temperature time evolution:

$$\frac{3}{2}\frac{d(n_{\mathrm{e}}T_{\mathrm{e}})}{dt} = P_{\mathrm{oh}} - \left(P_{\mathrm{ion}} + P_{\mathrm{line}} + P_{\mathrm{prb}}\right) \tag{99}$$

Rate equation for the plasma density time evolution:

$$\frac{dn_{\mathrm{e}}}{dt} = R_{\mathrm{ion}}(T_{\mathrm{e}}, n_{\mathrm{e}})n_0 n_{\mathrm{e}} - R_{\mathrm{rec}}(T_{\mathrm{e}}, n_{\mathrm{e}})n_{\mathrm{e}}n_{\mathrm{i}}. \tag{100}$$

5.2. Calculation of the plasma current

After getting the equations for calculating plasma density and plasma temperature in Section 5.1, we can compute the plasma parameters at each time step using the previously established equations with the initial conditions, including the initial plasma temperature, experimental gas pressure, and central solenoid current profile.

It is important to note that with higher temperature, the Lamor radius becomes too large such that ions cannot be confined. Although the expected operating temperature of the mini-Tokamak is limited to approximately 10 eV due to the gyro-radius constraint discussed in Appendix A.1, this section intentionally ignores that confinement limit. This is because low-

temperature conditions, while physically realistic, often result in low ionization fraction and minimal dynamic changes, which hinder the ability to assess the model's behavior and performance across a wide range of parameters. Therefore, in this section, we allow the temperature to rise freely and focus on verifying the model's internal consistency, specifically the evolution of plasma current, temperature, input and loss power, and energy conservation. Once the model has been validated under these conditions, the following section will incorporate the confinement limit and present a physically feasible central solenoid current profile tailored to the mini-Tokamak's operational constraints.

In our simulations, the initial plasma temperature is set to 0.026 eV (approximately 300 K), representing room temperature conditions. The working gas is He. The total neutral density is taken to be 10^{17} m^{-3}, which corresponds to a gas pressure around 10^{-5} Torr. The applied central solenoid current profile, shown in the blue solid line in Figure 33(a), linearly ramps down from 10 kA to 0 kA over the first 10 ms, followed by a ramp from 0 to −5 kA over the next 40 ms. The results for each case are presented in Figure 33 to Figure 36, illustrating how plasma parameters evolve over time under these conditions.

Figure 33(a) reveals that the plasma current begins to increase after about 4 ms, eventually reaching approximately 18 kA by the end of the 40 ms simulation, which reaches

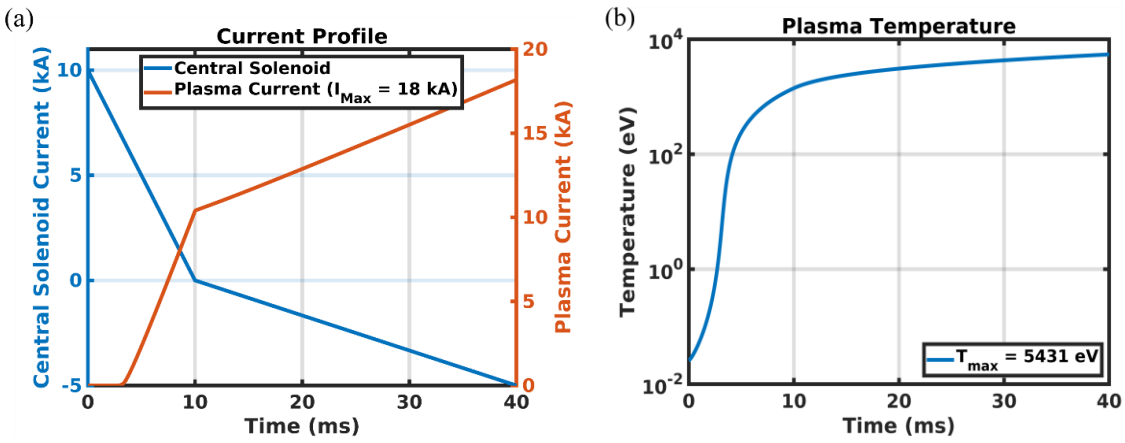

Figure 33: (a) Central solenoid and plasma current. (b) Plasma temperature.

our intended target of 10 kA. Meanwhile, Figure 33(b) shows that the plasma temperature climbs sharply after 4 ms and ultimately reaches over 5400 eV, far exceeding the confinement-safe regime, which is 10 eV.

This behavior can be understood by considering the dynamics of plasma resistance and heating. At the beginning of the simulation, when the electron temperature remains at 0.026 eV, the plasma resistivity is extremely high because resistivity scales as $\eta \propto T_e^{-1.5}$, as shown in Figure 34(a). According to Ohm's law (V = IR), high resistance severely limits the amount of current that can be induced in the plasma, despite the relatively strong voltage generated by the initial rapid change in solenoid current. As a result, the Ohmic heating power, given by $P_{\text{ohmic}} = I_p^2 R_p$, remains negligible, as shown in the blue solid line in Figure 34 (b). This explains why the plasma parameters stays nearly flat during the first 4 ms. Only when plasma current begins to rise after the plasma resistance drops to a sufficiently low level. As shown in Figure 33(a), the plasma current starts to increase. Correspondingly, other plasma parameters, such as temperature, ionization fraction, and Ohmic heating power, also begin to increase significantly from this point onward.

According to Eq. (76) and Eq. (78), plasma resistance R_p, which is shown in Figure 34 (a), is proportional to the Spitzer resistivity η_{Sp} and inversely proportional to the plasma temperature $T_e^{1.5}$:

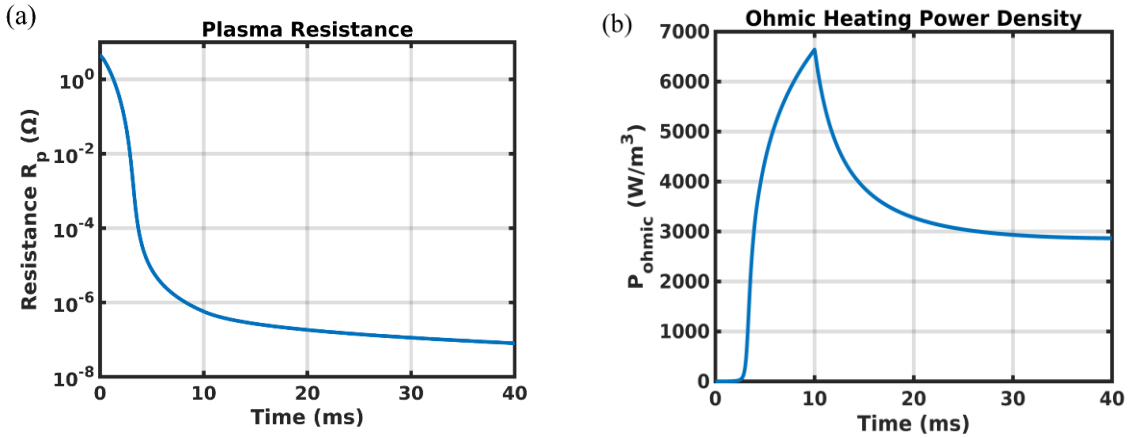

Figure 34: (a) Plasma resistance. (b) Ohmic heating power density.

$$R_{\mathrm{p}} \propto \eta_{\mathrm{Sp}} \propto T_e^{-1.5}. \tag{101}$$

It explains the exponential drop in resistance as temperature rises. Initially, with $T_{\mathrm{e}} = 0.026$ eV, the resistance is around 0.5 Ω with the assumption that breakdown occurred $t = 0$. However, by 50 ms, it falls below 10^{-7} Ω.

Figure 35(a) shows that the ionization fraction γ reaches full ionization ($\gamma \approx 1$) around 8 ms. The effect of ionization is also evident in the energy loss analysis shown in Figure 35(b). Ionization loss P_{ion}, corresponding to the orange solid line, begins to rise significantly around 3 ms and peaks near 5 ms, when the plasma is rapidly transitioning from weakly ionized to partially ionized. This behavior reflects the underlying physics of ionization loss, which scales as $P_{\mathrm{ion}} \propto n_0 n_{\mathrm{e}}$. In the early stage, although the neutral density n_0 is high, the electron density n_{e} is still too low to drive substantial ionization. As the temperature increases and electron density rises, the ionization fraction accelerates sharply, leading to a rapid increase in energy loss. After full ionization is achieved around 8 ms, $n_0 \to 0$, and ionization loss quickly drops to negligible levels.

Following this, the dominant power loss mechanism shifts to line radiation P_{line}, shown in red solid line in Figure 35(b). Line radiation arises from bound–bound transitions in ions, when electrons in excited states fall to lower energy levels, emitting photons with

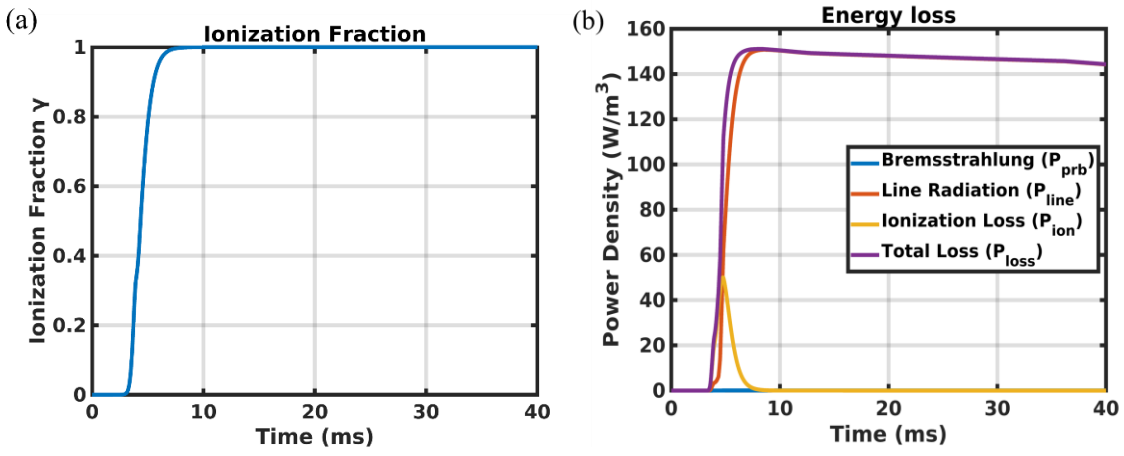

Figure 35: (a) Ionization fraction. (b) Energy loss.

specific energies. This radiation mechanism depends on both electron and ion densities ($P_{\mathrm{line}} \propto n_{\mathrm{e}} n_{\mathrm{i}}$) and remains significant throughout the rest of the simulation.

In contrast, Bremsstrahlung radiation P_{prb}, which originates from the deceleration of electrons in the electric fields of ions, remains negligible throughout the simulation, as shown by the blue line in Figure 35(b). This is primarily because in our assumption plasma consists of singly charged helium ions $(Z = 1)$, for which Bremsstrahlung losses are low. As a result, its contribution is several orders of magnitude smaller than ionization and line radiation and can be neglected.

In the present simulation, we assume $Z = 1$ for simplicity. This assumption is reasonable because the mini-Tokamak operates at relatively low temperatures (below 10 eV), where only the first ionization stage of helium is significantly populated. Moreover, the FIRST device primarily uses hydrogen as the working gas, which naturally has $Z = 1$, making the same assumption applicable.

However, if future designs aim to push the mini-Tokamak to higher temperatures, or if other gases such as helium or argon are used in devices like FIRST, then the assumption of $Z = 1$ will no longer hold. Higher temperatures can lead to multiple ionization stages (e.g., He^{2+}, Ar^{3+} and beyond), and alter both the charge state distribution and the associated radiation losses. In such cases, line radiations from both higher-Z ions and Bremsstrahlung, scale more strongly with Z and, would play a more prominent role. Therefore, a more detailed treatment of ionization stages and the effective charge Z_{eff} will be necessary to accurately capture energy loss and transport in future models.

Finally, Figure 36 demonstrates that the energy balance is well preserved in the simulation. The Ohmic heating from the central solenoid serves as the energy input to the

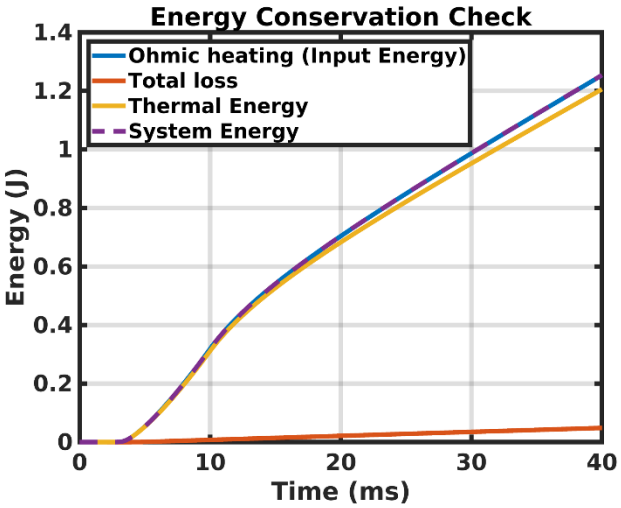

Figure 36: Energy conservation check.

plasma, which is the blue solid line. To calculate the total energy input, we first convert Eq. (92) into total input power (W) by multiplying by the plasma volume (V_p):

$$P_\mathrm{oh} = \frac{I_\mathrm{p}^2 R_\mathrm{p}}{V_\mathrm{p}} \left(\mathrm{W}/\mathrm{m}^3\right) \Rightarrow P_\mathrm{oh} V_\mathrm{p} \ (\mathrm{W}) \tag{102}$$

where I_p is plasma current (A), R_p is plasma resistance (Ω), and V_p is plasma volume (m^3). Next, to determine the total accumulated energy input over time, we integrate P_oh from $t = 0$ to the current time $t = 40$ ms:

$$E_\mathrm{input} = V_\mathrm{p} \int_0^t P_\mathrm{oh} dt \approx V_\mathrm{p} \sum_{i=1}^{n} P_\mathrm{oh}(i) \cdot \Delta t \tag{103}$$

where E_input is the total accumulated energy input in Joule to the plasma over time, $P_\mathrm{oh} V_\mathrm{p}$ is the Ohmic heating power in Watt, and Δt is the time step used for numerical integration.

The total energy loss of the plasma, as shown in the red solid line in Figure 36, is composed of Bremsstrahlung radiation (P_prb), line radiation (P_line), and ionization loss (P_ion). The ionization loss corresponds to Eq. (94) and accounts for the energy required to ionize neutral atoms through electron-neutral collisions. The radiation loss corresponds to Eq. (95) and represents the energy emitted as photons when excited ions return to lower energy states via bound–bound transitions. In addition, the Bremsstrahlung radiation loss

corresponds to Eq. (96) and arises from the deceleration of electrons in the Coulomb field of ions, producing broadband photon emission.

The net energy retained by the plasma is given by the equation:

$$U = \frac{3}{2} n_{\mathrm{e}} \cdot V_{\mathrm{p}} \cdot T_{\mathrm{e}} \times (\mathrm{eV\_to\_J}) \tag{104}$$

where U represents the internal energy of the plasma, measured in joules (J), which quantifies the total thermal energy stored within the plasma. The variable V_{p} represents the plasma volume (m^3). The term T_{e} represents the plasma temperature, measured in electron volts (eV), which describes the thermal energy per particle in the plasma. The final term, $\mathrm{eV\_to\_J}$, is a conversion factor from electron volts to joules, with a value of 1.6×10^{-19} J/eV.

As shown in Figure 36, the input energy from Ohmic heating is equal to the total system energy, which includes thermal energy and energy losses. This balance ensures that the program correctly accounts for energy input, dissipation, and conversion, thereby confirming that energy conservation is maintained within the model.

5.3. Sensitivity test of the program to the initial temperature

Before utilizing the program in Subsection 5.2 to further refine the design of the central solenoid current profile, we first need to test the program's sensitivity to the initial temperature. This verification is important because the initial temperature serves as a critical starting condition for plasma evolution. Since the governing equations for plasma density, temperature, and resistance all involve temperature-dependent terms, variations in the initial temperature could impact the overall plasma dynamics. By testing the program's response to different initial temperatures, we can ensure that the numerical simulation remains stable and consistent, regardless of the initial condition chosen.

This sensitivity test allows us to evaluate whether small variations in the initial temperature lead to significantly different plasma behavior, which could indicate potential

uncertainty in the model. If the simulation remains consistent under different initial temperature conditions, we can confidently proceed to use the program for further optimization of the central solenoid current profile. We performed calculations under different initial temperature conditions, setting T_0 = 0.026 eV, 5 eV, and 20 eV, as shown in Figure 37(a) and (b).

As observed in Figure 37(a), when the initial plasma temperature is relatively high (e.g., 20 eV), it first decreases before rising. This occurs because at higher temperature, energy losses due to line radiation, ionization, or radiation dissipation can exceed Ohmic heating. Consequently, the net power, defined as $P_{\text{net}} = P_{\text{oh}} - P_{\text{loss}}$, may initially become negative, leading to a drop in temperature. Over time, as electron density and plasma resistance evolve, Ohmic heating increases, eventually surpassing the losses and driving the temperature back up, resulting in the observed downward-then-upward trend.

Additionally, this initial temperature drop reduces the ionization fraction, as shown in Figure 37(b), since ionization is strongly temperature dependent. Ultimately, as the current and resistance evolve, Ohmic heating "catches up" with energy losses, stabilizing and increasing the temperature again.

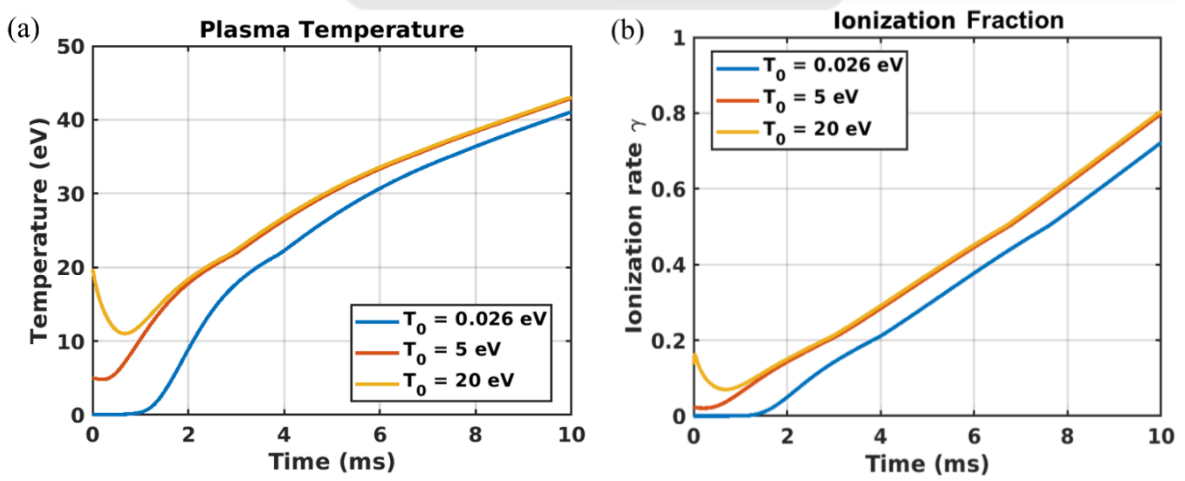

Figure 37: (a) Temporal evolution of plasma temperature for different initial temperature. (b) Temporal evolution of ionization fraction for different initial temperatures

The plasma temperature and ionization fraction result for the three different initial temperatures ultimately converge. This convergence occurs because the net heating power and energy loss in the plasma reaches a dynamic balance over time, driving the system toward a similar final temperature and ionization fraction regardless of the initial condition.

This behavior can be attributed to several mechanisms. Plasma resistance decreases as the temperature rises, particularly with higher electron temperatures. At higher temperatures, the lower resistance reduces the effectiveness of Ohmic heating ($P_{\mathrm{oh}} = I_{\mathrm{p}}^2 R_{\mathrm{p}} / V_{\mathrm{p}}$), while at lower initial temperatures, higher resistance leads to stronger Ohmic heating, accelerating the temperature increase in the early stages.

Additionally, energy loss mechanisms, such as line radiation loss, become more significant at higher temperatures. For higher initial temperatures, these losses can temporarily exceed Ohmic heating, resulting in a negative net power ($P_{\mathrm{net}} = P_{\mathrm{oh}} - P_{\mathrm{loss}}$) and causing a temperature drop, as shown in the yellow line for 20 eV case in Figure 37(a). In contrast, for lower initial temperatures, Ohmic heating dominates over losses, leading to a faster temperature rise.

As plasma evolves, a balance is eventually established between Ohmic heating and energy losses. This equilibrium determines the final temperature range and is primarily influenced by the system's injected power, geometry, density, and radiative properties, rather than the initial temperature. Therefore, whether the plasma starts at 0.026 eV, 5 eV, or 20 eV, the temperature trajectories converge, with differences only in the early transitional behavior. Over longer timescales, the equilibrium between Ohmic heating and energy loss mechanisms governs the plasma's steady-state temperature.

In conclusion, through this calculation, we have demonstrated that our program has low sensitivity to the initial plasma temperature. This reliability allows us to proceed further refining the design of the central solenoid current profile using the program.

5.4. Neoclassical resistivity

In a toroidal magnetic field, not all charged particles can freely circulate around the torus. This spatial variation in magnetic field strength resembles the behavior of a magnetic mirror, causing certain particles to reflect back and become confined between two strong-field regions. These so-called "trapped particles" cannot contribute effectively to the toroidal current, as they oscillate locally rather than circulating around the entire torus. This effect is particularly important for electrons, some of them are trapped and enter so-called banana orbits due to the mirror-like field structure. As a result, only a fraction of the total electron population participates in current conduction, leading to a reduction in the overall plasma conductivity. Consequently, this reduced efficiency must be accounted for estimating resistivity and heating performance in tokamak plasmas.

To account for these effects, neoclassical theory extends the classical model by incorporating the geometry of the tokamak and the behavior of trapped particles. It modifies the effective conductivity and resistivity of the plasma depending on the degree of collisionality and magnetic geometry. Neoclassical resistivity becomes important when a significant fraction of particles is trapped, which is often the case in compact or low-aspect-ratio devices like the mini-Tokamak.

In this study, we employ the neoclassical model developed by O. Sauter et al., published in *Physics of Plasmas* (1999)[18], which provides a set of widely used analytical formulas to compute key transport quantities such as the collisionality, conductivity, and bootstrap current. The first step involves evaluating the electron collisionality (ν_e^*), which characterizes the ratio between the electron collision frequency and the bounce frequency of trapped particles. The electron collisionality:

$$\nu_e^* \propto \frac{qRn_e Z_{\text{eff}} \ln \Lambda}{T_e^2 \epsilon^{1.5}}, \tag{105}$$

where q is the safety factor, R is the major radius of plasma, ϵ is the inverse aspect ratio ($\epsilon = a/R$), Z_{eff} is the effective ion charge, which is assumed to be 1 in our assumption, and $\ln \Lambda$ is the Coulomb logarithm. A high ν_e^* indicates strong collisionality and classical transport behavior, whereas a low ν_e^* suggests that neoclassical effect dominates due to the increased influence of trapped particles.

The safety factor q describes the pitch angle of magnetic field lines in a tokamak and is defined as the number of toroidal turns a magnetic field line makes for each poloidal turn. It plays a crucial role in determining the stability and transport properties of the plasma. Notice that the pitch angle of magnetic field lines may be different at different radial distance from the plasma center axis. In Eq. (106), the safety factor appears in the numerator, meaning that higher q values increase the collisionality. To evaluate $q(r)$, we use the following physics-based relation derived from the magnetic field geometry in a tokamak:

$$q(r) = \frac{rB_T}{(R + r)B_p} \tag{106}$$

where r represents the radial distance from the plasma center axis, while R is the major radius of plasma, which is 85 mm for mini-Tokamak. B_T is the toroidal magnetic field, and B_p is the poloidal magnetic field.

The toroidal magnetic field B_T is assumed to vary with radius according to the simple model:

$$B_T(r) = B_0 \frac{R}{R + r} \tag{107}$$

where $B_0 = 0.1\ \text{T}$ is the magnetic field measured at the reference radius of $R = 85\ \text{mm}$. This radial dependence reflects the geometric decay of toroidal field strength as the radius increases.

The poloidal magnetic field B_p is derived from Ampère's law under the assumption of uniform current distribution within a given radius r. The local poloidal current density is expressed as:

$$J_{\mathrm{p}} = \frac{I_{\mathrm{p}}}{\pi r^2} \,. \tag{108}$$

Applying Ampère's circuital law over a circular loop of radius r, we obtain:

$$\oint B_{\mathrm{p}} \cdot dl = 2\pi r \cdot B_{\mathrm{p}} = \mu_0 I_{\mathrm{p}} = \mu_0 \pi r^2 J_{\mathrm{p}} \,, \tag{109}$$

which leads to:

$$B_p = \frac{\mu_0 r}{2} J_{\mathrm{p}} \,. \tag{110}$$

Substituting Eq. (107) and Eq. (110) into Eq. (106):

$$q(r) = \frac{r}{R+r} \times \frac{B_0 R}{R+r} \times \frac{1}{\frac{\mu_0 r}{2} J_{\mathrm{p}}} = \frac{2B_0 R}{\mu_0 J_{\mathrm{p}} (R+r)^2} \,. \tag{111}$$

After calculating the local safety factor profile $q(r)$ across the plasma radius, the average safety factor $\bar{q}$ is computed using a radial weighting:

$$\bar{q} = \frac{\int_0^a q(r) \cdot r \, dr}{\int_0^a r \, dr} = \frac{2}{a^2} \int_0^a q(r) \cdot r \, dr \,, \tag{112}$$

where a is the minor radius of plasma, which is 55 mm for mini-Tokamak. Although the plasma cross-section is elliptical, the weighting remains proportional to r because each magnetic surface is assumed to be a geometrically similar ellipse, causing the elliptical area factors to cancel out in both numerator and denominator. This average $\bar{q}$ is then used in Eq. (105) as q, as shown in the red dash line in Figure 38.

Next, the fraction of trapped particles is estimated. These particles become trapped due to magnetic field inhomogeneity and are influenced by both the aspect ratio and triangularity of the plasma cross-section. A refined formula incorporating triangularity corrections is employed to compute f_{t}:

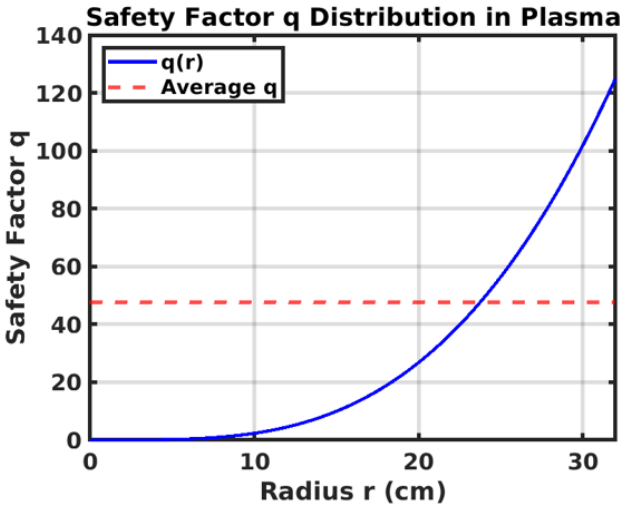

Figure 38: Radial profile of safety factor.

$$f_{\mathrm{t}} = 1 - \frac{1-\epsilon_{eff}}{1+2\sqrt{\epsilon_{eff}}} \cdot \sqrt{\frac{1-\epsilon}{1+\epsilon}}. \tag{113}$$

where ϵ_{eff} is the triangularity corrections:

$$\epsilon_{\mathrm{eff}} = 0.67(1 - 1.4\delta^2) \cdot \epsilon. \tag{114}$$

Here, δ is the plasma triangularity, which characterizes the D-shape deformation of the plasma boundary. However, in our present calculation, the plasma is assumed to have an elliptical cross-section without triangular shaping. Therefore, we set $\delta = 0$, neglecting triangularity effects in the estimation of the trapped particle fraction. Since trapped particles are confined by mirror-like magnetic effects, they contribute less effectively to the toroidal current. As f_{t} increases, the effective conductivity decreases. This behavior is especially important in low-aspect-ratio configurations like mini-Tokamak, where geometric effects are more pronounced. This correction reduces the influence of ϵ in plasmas with non-circular cross-sections. Using ϵ_{eff} instead of ϵ allows for a more accurate estimate of the trapped particle fraction, especially in cases with significant triangular shaping. In our case, since $\delta = 0$, we recover $\epsilon_{\mathrm{eff}} = 0.67 \cdot \epsilon$.

The neoclassical conductivity σ_{neo} is then obtained by correcting the classical Spitzer conductivity σ_{spitzer} using a term that accounts for trapped particles:

$$\sigma_{\text{neo}} = \sigma_{\text{spitzer}} \cdot \left[1 - f_{33}^{eff}\left(1 + \frac{0.36}{Z_{\text{eff}}} - f_{33}^{eff}\left(\frac{0.59}{Z_{\text{eff}}} - \frac{0.23}{Z_{\text{eff}}} f_{33}^{eff}\right)\right)\right]. \tag{115}$$

Here, f_{33}^{eff} is an effective trapped particle fraction, defined as:

$$f_{33}^{eff} = \frac{f_{\text{t}}}{1 + (0.55 - 0.1f_t)\sqrt{v^*} + -0.45(1 - f_t) \cdot \frac{v^*}{Z_{eff}^{1.5}}}. \tag{116}$$

This effective correction factor is used instead of the raw trapped particle fraction f_{t} to account for the influence of electron collisionality (v^*) and impurity content (Z_{eff}) on the actual current-carrying capability of the trapped particle population. While f_{t} captures the geometric tendency for particles to become trapped due to magnetic field inhomogeneity, not all trapped particles equally suppress conductivity. At low collisionality, these particles remain trapped and significantly hinder current flow. At higher collisionality, frequent collisions allow them to scatter into passing orbits, reducing their suppressive effect. Therefore, f_{33}^{eff} serves as a refined measure of the effective transport-limiting role of trapped particles under realistic tokamak conditions.

From Eq. (116) and Eq. (117), we observe that in the limit where the trapped particle fraction $f_{\text{t}} \to 0$, the effective correction $f_{33}^{eff} \to 0$, and thus the neoclassical conductivity σ_{neo} approaches the classical Spitzer conductivity σ_{spitzer}, as expected for a fully passing particle population. Conversely, when f_{t} approaches unity and the collisionality v^* remains moderate, the value of f_{33}^{eff} becomes significant. This leads to a strong suppression of σ_{neo}, reflecting the fact that trapped particles contribute much less to the net toroidal current, due to their limited ability to complete toroidal orbits.

The effective neoclassical resistivity is defined as:

$$\eta_{\text{neo}} = \frac{1}{\sigma_{\text{neo}}} . \tag{117}$$

This resistivity is then used to calculate the total plasma resistance via:

$$R_{\text{p}} = \eta_{\text{neo}} \frac{L}{A} = \eta_{\text{neo}} \frac{2\pi R}{\pi ab} \tag{118}$$

where $L = 2\pi R$ is the effective length of the plasma current path, which is the length of the plasma coil of the blue toroidal shape single-coil shown in Figure 30 with $R = 85$ mm as the major radius of plasma. A is the cross-sectional area of the plasma column given by $A = \pi ab$. The parameters $a = 55$ mm and $b = 100.1$ mm represent the minor and major axes of the plasma cross-section, respectively.

By applying this neoclassical framework, we capture key physical mechanisms that the classical model omits, such as magnetic geometry and trapped particle effects. Then we can compare plasma resistivity and temperature evolution using both Spitzer and neoclassical models to highlight the importance of this correction.

The initial conditions are identical to those described in Section 5.2. The plasma temperature is set to 0.026 eV, and the working gas is helium (He). The total neutral density is taken to be 10^{17} m^{-3}. The applied central solenoid current profile, shown as the blue solid line in Figure 39(a), linearly ramps down from 10 kA to 0 kA over the first 10 ms, followed by a ramp from 0 kA to −5 kA over the subsequent 30 ms. A comparison between the results

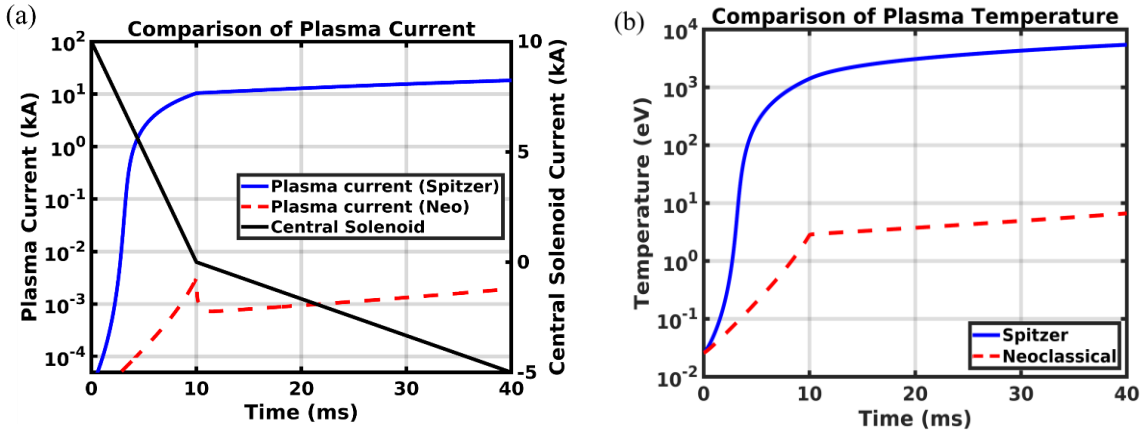

Figure 39: (a) Central solenoid and plasma current. (b) Plasma temperature.

obtained using neoclassical and Spitzer resistivity models for plasma parameters is presented in Figure 39 and Figure 40.

As shown in Figure 39(a), when the same central solenoid current is applied, which is the black solid line, the plasma current evolution differs significantly between the two models. Under the Spitzer model, corresponding to the blue solid line, plasma current quickly rises and reaches a steady state of over 10 kA within the first 10 ms. In contrast, the neoclassical model predicts a peak plasma current of only around 3.2 A. Subsequently, as the solenoid current change rate decreases from 10 kA over 10 ms to 5 kA over 30 ms, the induced plasma current also declines. This reduction weakened the Ohmic heating, which in turn slowed down the increase in plasma temperature, as shown in Figure 39(b). The resulting temperature stagnation leads to a rise in plasma resistance, as illustrated in Figure 40(a). According to Ohm's law $V = IR$, for a given induced voltage, an increase in plasma resistance leads to a reduction in plasma current. Moreover, the Ohmic heating power, which determines plasma temperature evolution, is given by $P_{\mathrm{ohmic}} = I_{\mathrm{p}}^2 R_{\mathrm{p}}$. This relationship introduces a trade-off: increasing plasma resistance tends to increase in Ohmic heating power, but simultaneously reducing plasma current leads to a potentially greater reduction in Ohmic heating power overall.

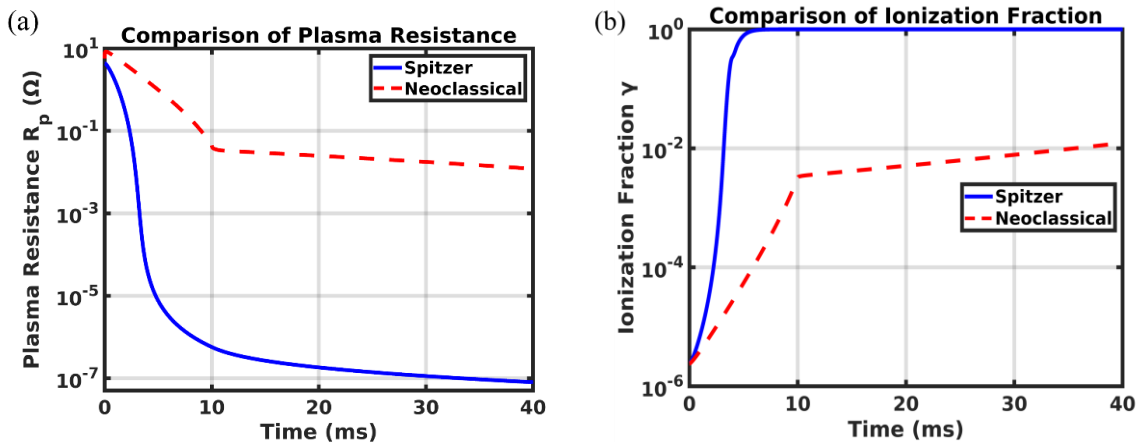

Figure 40: (a) Plasma resistance. (b) Ionization fraction.

In this simulation, that trade-off clearly favors current suppression. As seen in Figure 39(a), the plasma current in the Spitzer case reaches it peaks at only around 18 kA by 40 ms, whereas in the neoclassical case it peaks at only around 3.2 A, a ratio of nearly 5.6×10^3. At the same time, Figure 40(a) shows that the Spitzer resistance is about 10^{-7} Ω while the neoclassical resistance is approximately 10^{-2} Ω. At that moment, it yields a resistance ratio of roughly 10^{-5}. Despite the higher resistance in the neoclassical case, the suppression in current is more dominant, leading to a substantial reduction in Ohmic heating.

Figure 39(b) compares the evolution of plasma temperature. With the Spitzer model, the plasma temperature rapidly increases to nearly 10^4 eV due to efficient Ohmic heating. In contrast, the neoclassical model yields much slower temperature growth, reaching less than 10 eV at t = 40 ms.

Figure 40(a) shows the time evolution of plasma resistance. The Spitzer model predicts a steep decrease in resistance as the temperature increases ($R_\mathrm{p} \propto T_\mathrm{e}^{-1.5}$), reaching values below 10^{-7} Ω. In contrast, the neoclassical model shows a much more gradual decline, with resistance stabilizing around 10^{-2} Ω, consistent with reduced conductivity due to trapped particle effects.

Figure 40(b) shows a similar trend in ionization behavior. Under the Spitzer model, the ionization fraction γ quickly rises and reaches full ionization ($\gamma \approx 1$) by around 8 ms.

However, in the neoclassical model, the ionization fraction reaches only 1% at the final time step (t = 40 ms). This difference arises because neoclassical resistivity leads to a lower plasma temperature, and the ionization fraction in our model depends strongly on temperature as shown in Eq. (80) and Eq. (89).

This comparison reveals that relying solely on the Spitzer model can significantly overestimate both Ohmic heating efficiency and plasma current drive during the startup phase, potentially leading to unrealistic expectations for successful plasma initiation. In

contrast, the neoclassical model provides a more accurate description of transport behavior, particularly in low-aspect-ratio configurations such as our mini-Tokamak, where small geometric effects strongly influence trapped particle dynamics and reduce effective conductivity.

Accordingly, in the following section, we employ neoclassical calculations to design the central solenoid current profile, aiming to achieve a more realistic simulation of plasma startup, current evolution, and temperature rise.

5.5. Calculation results

The solenoid current profile in the mini-Tokamak was designed with the goal of achieving a plasma temperature of 10 eV and a plasma current of 10 kA. In our model, the plasma current is induced by the changing magnetic flux and is therefore highly dependent on the rate of change of the central solenoid current. However, driving a high plasma current requires a rapid variation of the solenoid current, which in turn results in stronger Ohmic heating. This rapid heating can cause the plasma temperature to exceed the 10 eV limitation, beyond which He^+ ions may not remain confined. Such a loss of confinement could potentially lead to serious damage to the vacuum-vessel wall or diagnostic components, due to energetic particle impacts. To prevent this, the current solenoid profile is designed primarily to regulate the plasma temperature, ensuring it reaches and maintains around 10 eV. As a result, the target plasma current of 10 kA is not prioritized in present design.

Three cases were considered: (a) Gas density of 10^{17} m^{-3}, with initial temperature of 0.026 eV; (b) Gas density of 10^{17} m^{-3}, with initial temperature of 1 eV; (c) Gas density of 10^{18} m^{-3}, with initial temperature of 1 eV. All cases are listed in Table 3. The detailed code for plasma parameters calculation is provided in Appendix A.14.

Table 3: Plasma startup scenarios and central solenoid current profiles in mini-Tokamak.

Case	n_0	T_0	$I_{cs\_max,1}$	Δt_{cs_1}	$I_{cs\_max,2}$	Δt_{cs_2}	V_{loop_1}	V_{loop_2}
a	10^{17} m^{-3}	0.026 eV	10 kA	10 ms	-7.5 kA	40 ms	0.16 V	0.03 V
b	10^{17} m^{-3}	1 eV	5.4 kA	10 ms	-1.5 kA	40 ms	0.086 V	0.006 V
c	10^{18} m^{-3}	1 eV	10 kA	10 ms	-10 kA	10 ms	0.16 V	0.16 V

(a) The initial plasma temperature is set to 0.026 eV, and the gas density is assumed to be 10^{17} m^{-3}, corresponding to a pressure of approximately 10^{-5} Torr, the results are shown in Figure 41(a). The applied central solenoid current profile, represented by the blue solid line in Figure 41(a), decreases linearly from $I_{cs\_max,1} = 10$ kA to 0 kA over the first 10 ms (Δt_1), followed by a linear ramp from 0 kA to $I_{cs\_max,2} = -7.5$ kA over the next 40 ms (Δt_2). This time-varying current induces a loop voltage of approximately 0.16 V over the first 10 ms, which exceeds the required breakdown voltage of 0.046 V at R = 85 mm under a gas pressure of 10^{-5} Torr, as calculated with Eq. (62) and Eq. (63). This confirms that the chosen current profile provides sufficient loop voltage to initiate plasma breakdown under the target conditions. Figure 41(b) shows the plasma temperature successfully reaches the target value of 10.9 eV. The plasma current, corresponding to the red solid line in Figure 41(a), peaks at only 4.2 A.

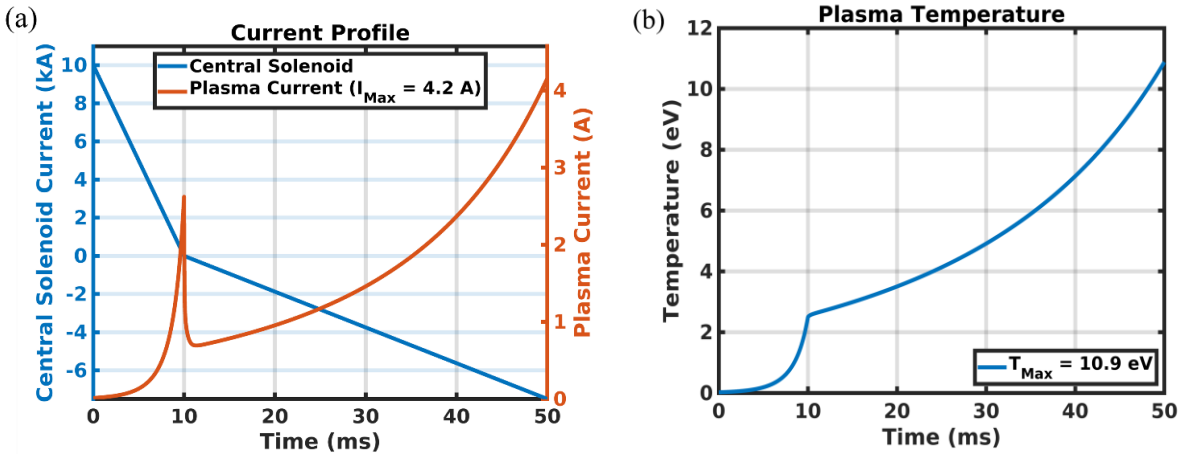

Figure 41: (a) Central solenoid and plasma current. (b) Plasma temperature.

Additionally, the ionization fraction remains low throughout the discharge, as shown in Figure 42(a), reaching only 2.63% by the end of the simulation. While Figure 42(b) shows the plasma resistance time evolution, which gradually decreases as the temperature increases.

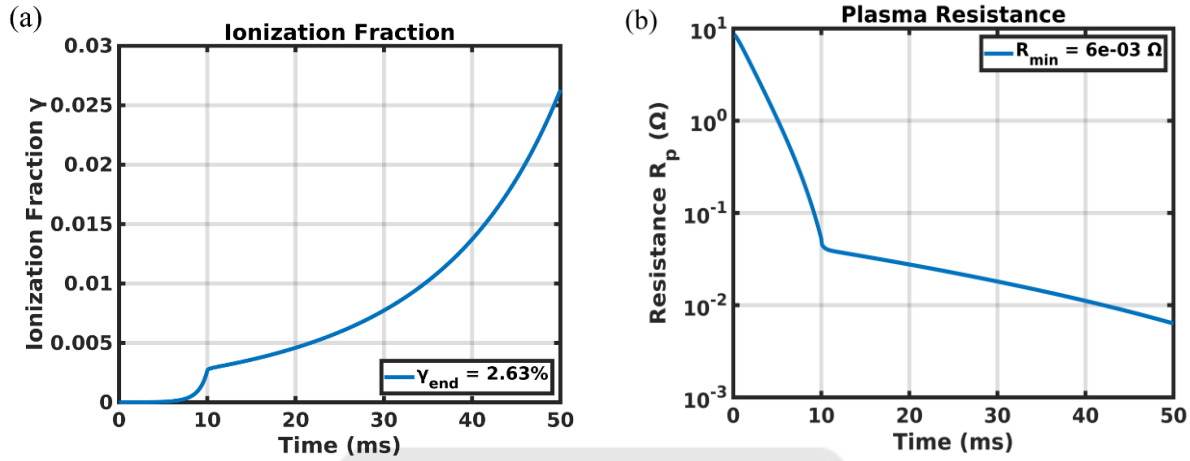

Figure 42: (a) Ionization fraction. (b) Plasma resistance.

(b) The initial plasma temperature is set to 1 eV, and the gas density is assumed to be 10^{17} m^{-3}, corresponding to a pressure of approximately 10^{-5} Torr, the results are shown in Figure 43(a). The applied central solenoid current profile, represented by the blue solid line in Figure 43(a), decreases linearly from 5.4 kA to 0 kA over the first 10 ms, followed by a linear ramp from 0 kA to −1.5 kA over the next 40 ms. This current profile induces a loop

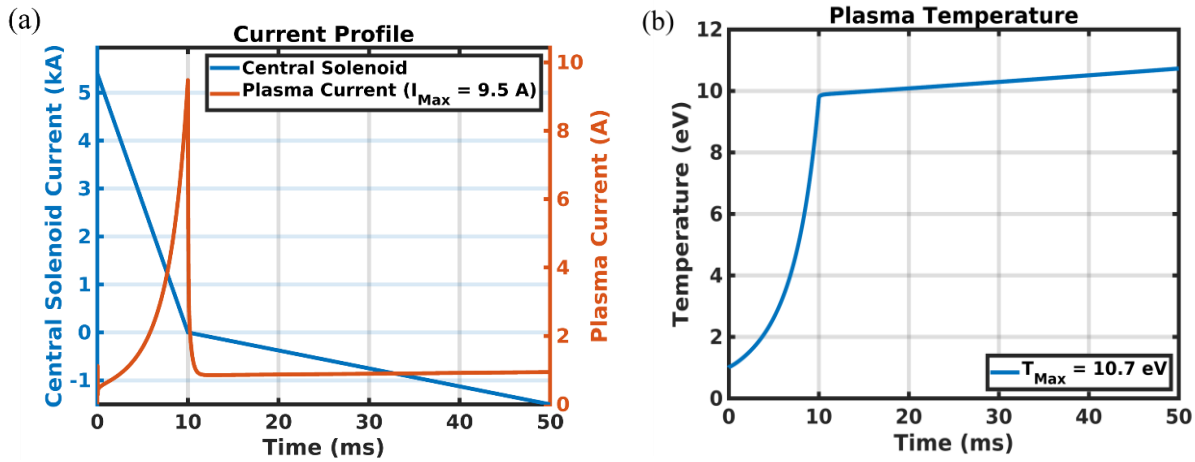

Figure 43: (a) Central solenoid and plasma current. (b) Plasma temperature.

voltage of approximately 0.086 V over the first 10 ms, which exceeds the required breakdown voltage of 0.046 V at R = 85 mm under a gas pressure of 10^{-5} Torr.

With the application of preheating, the plasma temperature can be rapidly raised to approximately 10 eV even under a reduced peak central solenoid current. It can then be maintained near this level for an extended period, despite a slower central solenoid current change rate, thereby providing a longer observation window for studying plasma behavior under stable conditions. However, the plasma current, represented by the red solid line in Figure 43(a), exhibits a different evolution. It initially rises to near 9.5 A at 10 ms, but then drops sharply to approximately 0.8 A, due to the reduced rate of change in the central solenoid current.

Meanwhile, both the ionization fraction and plasma resistance stabilize shortly after the temperature plateaus. As shown in Figure 44(a), the ionization fraction reaches 2.58% and remains steady, while the resistance shown in Figure 44(b) decreases rapidly in the early stage and flattens out at 6 mΩ, consistent with the temperature profile.

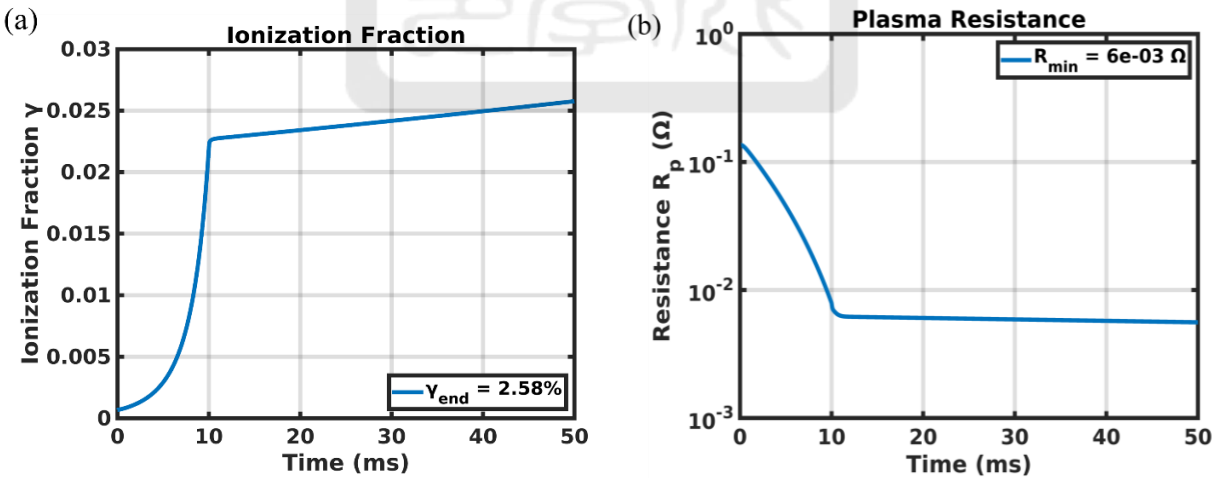

Figure 44: (a) Ionization fraction. (b) Plasma resistance.

(c) The initial plasma temperature is set to 1 eV, and the gas density is assumed to be 10^{18} m^{-3}, corresponding to a pressure of approximately 10^{-4} Torr, the results are shown in Figure 45(a). The applied central solenoid current profile, represented by the blue solid line in Figure 45(a), decreases linearly from 10 kA to 0 kA during the first 10 ms, followed by a further linear ramp from 0 kA to −10 kA over the next 10 ms. This current profile induces a loop voltage of approximately 0.16 V over the first 10 ms, which exceeds the required breakdown voltage of 0.066 V at R = 85 mm under a gas pressure of 10^{-4} Torr.

Figure 45(b) shows that, under this central solenoid current profile, the plasma temperature reaches a maximum of only 7.2 eV. This limitation arises from the capacitor-driven nature of the power supply, which requires that the energy associated with the rapidly decreasing solenoid current be dissipated elsewhere in the circuit, typically through external resistive elements. In our current hardware setup, the system is only capable of handling a linear decrease of the solenoid current from 10 kA to 0 kA within 10 ms, as a faster ramp could exceed the voltage or thermal limits of the components, posing a risk to system integrity. Although achieving the target plasma temperature requires a high current change rate to generate sufficient inductive electric fields for plasma initiation, we adopt this current profile as a compromise between effective plasma startup and the constraints of our existing

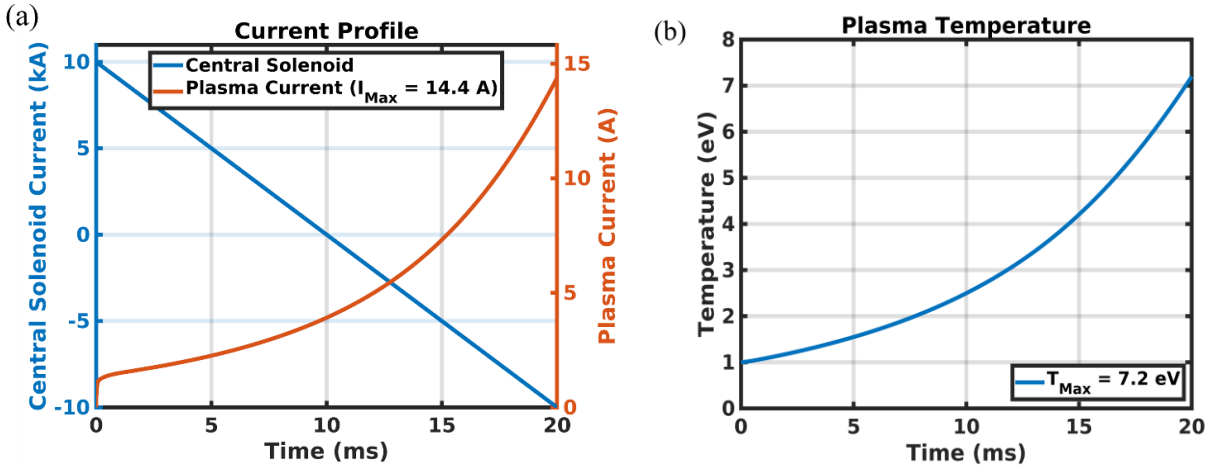

Figure 45: (a) Central solenoid and plasma current. (b) Plasma temperature.

hardware. Despite this temperature limit, the sharp current change still enables a strong inductive drive, pushing the plasma current to a peak value of approximately 14.4 A, as shown by the red solid line in Figure 45(a).

Figure 46(a) shows that, due to the relatively low plasma temperature, the ionization fraction reaches a maximum of 1.39%. Similarly, as shown in Figure 46(b), the plasma resistance remains relatively high, with a minimum value of approximately $10^{-2}\,\Omega$, also a result of the insufficient temperature rise.

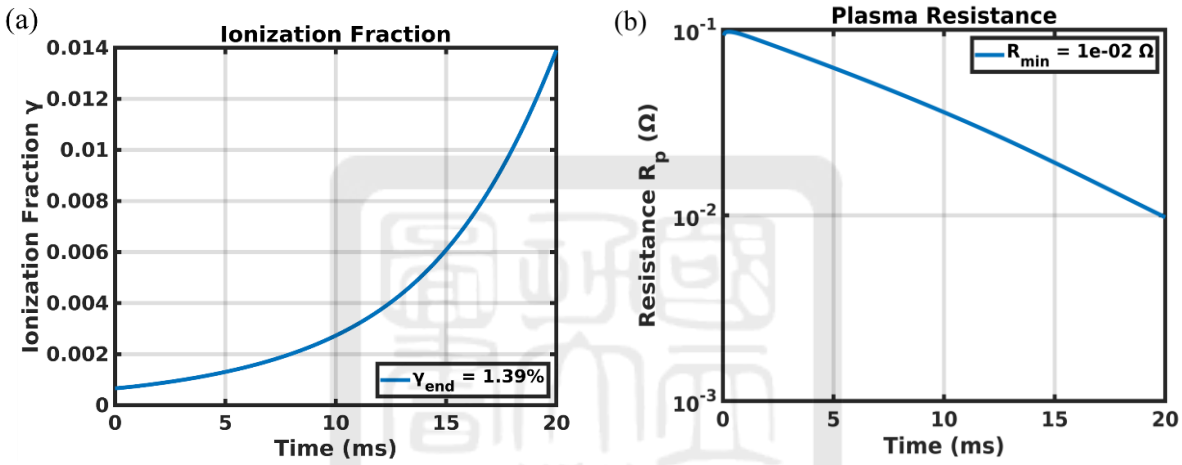

Figure 46: (a) Ionization fraction. (b) Plasma resistance.

In addition, the model is applied to the Formosa Integrated Research Spherical Tokamak (FIRST), the first tokamak being constructed in Taiwan, with detailed calculations provided in Appendix A.15.

5.6. Discussion

Our model primarily simulates the temporal evolution of plasma parameters during the plasma breakdown phase, influenced by Ohmic heating generated by the central solenoid current profile. In this simple model, physical mechanisms such as thermal conduction, particle diffusion, and transport were not included.

Specifically, for modeling electron density, this study employs a simplified rate equation:

$$\frac{dn_{\mathrm{e}}}{dt} = R_{\mathrm{ion}}(T_{\mathrm{e}}, n_{\mathrm{e}})n_0 n_{\mathrm{e}} - R_{\mathrm{rec}}(T_{\mathrm{e}}, n_{\mathrm{e}})n_{\mathrm{e}} n_{\mathrm{i}} - \nabla \cdot \Gamma. \tag{119}$$

In Eq. (80), only the first two terms in Eq. (119) were included, which account solely for local ionization and recombination reactions. The last term in Eq. (119), $\nabla \cdot \Gamma$, which was omitted in Eq. (80), represents density variations due to particle flow, diffusion, and plasma expansion or contraction, all of which are neglected in this simplified model. This simplification implicitly assumes that particle generation and loss occur solely through local reactions, with no spatial particle transport out of the simulation region. Consequently, this may introduce several limitations:

- Neglect of mass conservation and plasma flow effects: In actual plasmas, particles experience flow or expansion driven by electric fields or pressure gradients, phenomena that the simplified rate equation cannot represent.
- Inability to describe spatial density gradients and non-uniformities: Without a complete continuity equation, the spatial distribution of density near boundaries or regions with rapidly varying magnetic fields cannot be accurately depicted.
- Inability to predict plasma volume expansion or contraction: Ignoring changes in plasma volume may result in overestimated particle density and pressure, affecting predictions of temperature and energy dissipation.
- Reduced accuracy for long-term simulations: Accumulated errors from neglecting transport and flow become increasingly significant over extended simulation periods, thereby reducing the reliability of the model's predictions.

Furthermore, under ideal conditions, a strong magnetic field effectively confines plasma particles to magnetic field lines, primarily guiding particle motion along the field lines (parallel direction) and suppressing perpendicular (transverse) transport due to particle

gyration. In practice, however, magnetic confinement is imperfect, especially under conditions of low magnetic fields, non-stationary states, or inadequate boundary conditions, resulting in gradual particle escape from the plasma region. Common escape mechanisms include transverse diffusion caused by particle-neutral collisions, E×B drift, and turbulence-driven anomalous transport.

Since our model does not incorporate these transverse transport mechanisms, it effectively assumes ideal magnetic confinement, with particles permanently confined within the plasma region. This assumption can lead to the following inaccuracies:

- Overestimation of electron density: Ignoring particle loss due to transverse diffusion.
- Overestimation of electron temperature: High-energy electrons are more prone to escape; neglecting their loss underestimates energy dissipation.
- Misjudgment of plasma sustainment and stability conditions: Particularly in scenarios of poor confinement or proximity to vacuum-vessel walls, these errors become more pronounced, potentially resulting in overly optimistic predictions of steady-state conditions.

The energy balance equation is also affected by the lack of transport and particle loss mechanisms. To model the time evolution of electron temperature, this study adopts a simplified energy equation:

$$\frac{3}{2}\frac{d(n_e T_e)}{dt} = P_{oh} - \left(P_{ion} + P_{line} + P_{prb}\right) - P_{conv}\,. \tag{120}$$

In Eq. (91), only the first four terms in Eq. (120) were included. The last term in Eq. (120), $P_{conv} = 3n_e T_e / 2\tau_E$, accounting for convective energy losses, where τ_E is the energy confinement time, was omitted in Eq. (91). This term represents the energy carried away by escaping particles and may become the dominant loss mechanism under weak confinement conditions. Neglecting this term can lead to an overestimation of plasma temperature and an

underestimation of total power dissipation, which in turn may cause the required central solenoid current ramp rate for initiating breakdown to be underestimated.

Additionally, the model does not explicitly account for seed electron generation mechanisms. Typically, during the initial tokamak breakdown phase, neutral gas molecules dominate the environment, and stable plasma formation has not yet occurred. Initiating plasma formation requires a pre-existing population of free electrons (seed electrons) to undergo collisions with neutral atoms, triggering initial ionization and Townsend avalanche reactions that rapidly increase electron and ion densities, ultimately establishing stable plasma.

Seed electrons usually originate from external effects such as microwave-assisted ionization rather than from the applied electric field or Ohmic heating. Since our model omits detailed seed electron generation mechanisms and their production rates, it may underestimate the required electric field strength for plasma initiation and the actual delay time for breakdown, resulting in overly optimistic simulation outcomes.

Lastly, to more accurately reflect real plasma behavior, this study incorporates the neoclassical theory for calculating plasma resistivity. Compared to the traditional Spitzer theory, neoclassical theory is more suitable for tokamak, as it more precisely accounts for the influence of trapped particles formed by magnetic mirror effects on effective collision rates, conductivity, and resistivity. However, during the initial low-temperature breakdown phase, the electron kinetic energy is generally insufficient to form significant trapped orbits, and the velocity distribution remains approximately Maxwellian. Under such conditions, the collisionality, as defined in Eq. (105), becomes very high, and the neoclassical model naturally reduces to the traditional Spitzer theory, as shown in Eq. (115). In other words, although neoclassical theory is applied throughout, the inclusion of the collisionality parameter ν_e^* ensures that trapped-particle effects are suppressed when not physically

relevant, thereby avoiding an underestimation of plasma conductivity in the low-temperature regime. This feature allows the model to remain valid and accurate across both low-temperature and high-temperature regimes, and enables it to be directly extended to full-temperature tokamak operations, where reduced collisionality and trapped-particle effects become significant.

In summary, our model is based on several simplifying assumptions and is intended primarily as a first-order estimation tool for predicting breakdown parameters and supporting the preliminary design of the driving circuit. In the future, more comprehensive numerical simulations and experimental validation will be conducted to refine the predicted values and enhance the physical accuracy and reliability of the model.

5.7. Conclusion

In this section, we developed a model capable of calculating the evolution of various plasma parameters, including temperature, current, and resistance. We also tested the model's sensitivity to the initial plasma temperature and confirmed its robustness. By using this model, we can optimize the central solenoid current profile and refine hardware designs to ensure that the plasma temperature and current meet our experimental objectives. Table 4 presents the three selected scenarios used in this study where particle losses and the corresponding energy losses were neglected.

Table 4: Plasma startup scenarios and central solenoid current profiles in mini-Tokamak.

Case	n_0	T_0	$I_{\mathrm{cs\_max,1}}$	$\Delta t_{\mathrm{cs\_1}}$	$I_{\mathrm{cs\_max,2}}$	$\Delta t_{\mathrm{cs\_2}}$	$V_{\mathrm{loop\_1}}$	$V_{\mathrm{loop\_2}}$
a	10^{17} m^{-3}	0.026 eV	10 kA	10 ms	-7.5 kA	40 ms	0.16 V	0.03 V
b	10^{17} m^{-3}	1 eV	5.4 kA	10 ms	-1.5 kA	40 ms	0.086 V	0.006 V
c	10^{18} m^{-3}	1 eV	10 kA	10 ms	-10 kA	10 ms	0.16 V	0.16 V

Based on our results, case (b) is the most preferable scenario. In this case, the plasma temperature remains near 10 eV for a longer duration, providing an extended observation window for studying plasma behavior under quasi-steady conditions. Although case (c) yields a higher peak plasma current, our current focus is on achieving and sustaining sufficient electron temperature. Moreover, implementing the central solenoid current profile required for case (c) involves rapidly reducing the solenoid current from 10 kA to 0 kA within 10 ms, which may pose significant engineering and safety challenges. Therefore, case (b) is currently identified as the most practical and preferred scenario.

6. Future work

Our models in this study are designed for rapid estimation of plasma parameters and therefore involve several simplified assumptions. For example, we assume that the plasma adopts its final equilibrium shape immediately after breakdown, which was modeled as a single-turn elliptical torus, as illustrated in Figure 47.

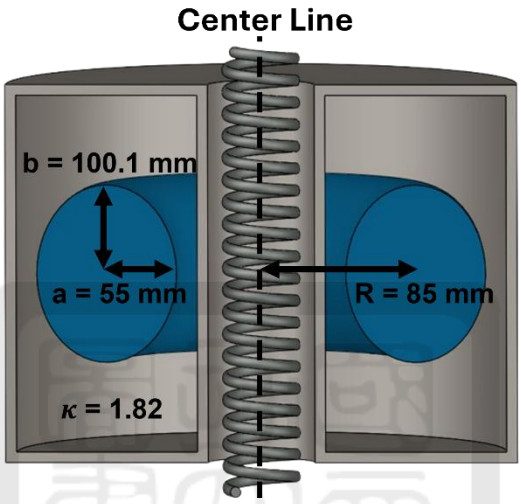

Figure 47: Simplified xz-plane cross-section of mini-Tokamak.

The breakdown voltage we compute in Section 4.1 corresponds to a breakdown occurring precisely at R = 85 mm. However, plasma tends to form initially near the inner vacuum-vessel wall, where the electric field is stronger under the same induced loop votlage. As shown in Figure 48(a), the plasma should be initiated near the inner wall and gradually expands outward.

In future work, we plan to incorporate this expansion behavior into our model by discretizing the plasma region into multiple radial grids, as illustrated in Figure 48(b). Each grid will be assigned its own breakdown voltage based on its position. If the induced loop voltage at a given grid exceeds the local breakdown threshold, that region will be treated as plasma and included as a coil components in the full circuit model in Chapter 3. Otherwise, it will remain as neutral gas.

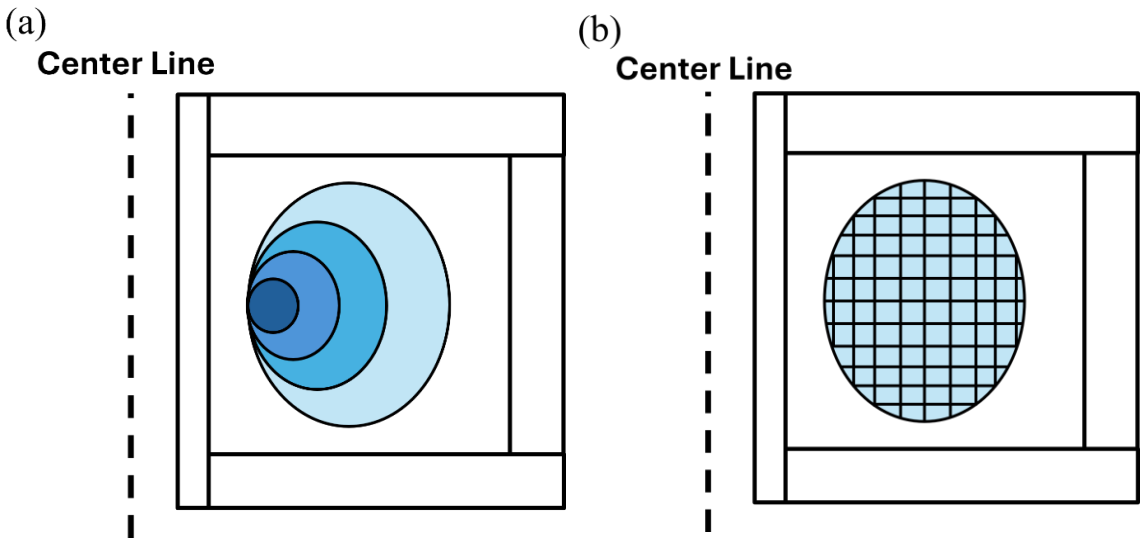

Figure 48: (a) Plasma expansion process. (b) Grid-based plasma discretization model.

Thermal conduction will be included between neighboring grids, allowing us to account for heat exchange between plasma and gas regions. A grid that is not initially broken down may eventually be defined as plasma if its temperature exceeds a threshold.

Using the model introduced in Chapter 3, we can compute the mutual inductive currents between all components, including those representing plasma grids. This allows us to determine the evolving plasma current distribution across space and time in a physically consistent way. Ultimately, this approach will enable more accurate modeling of plasma parameter evolution and support the optimization of the central solenoid current profile design.

In addition, particle losses and the corresponding energy losses, as described in Eq. (119) and Eq. (120) in Section 5.6, will be incorporated to provide a more physically consistent and accurate simulation of plasma behavior during the breakdown phase.

7. Conclusion

In this thesis, we developed a series of models to support the design and optimization of the central solenoid current profile in the mini-Tokamak, including the effects of eddy currents in the vacuum-vessel walls. This work includes several key physical modules, including the calculation of component resistance, self-inductance and the mutual inductance between all the components, the calculation of eddy currents induced by time-varying magnetic fields from central solenoid, and the time evolution of plasma parameters such as temperature, ionization fraction, current, and resistivity.

We began by establishing an accurate geometric model of the vacuum vessel. Based on this geometry, we first calculated the resistance and self-inductance of all components, as well as the mutual inductance between them. These quantities were assembled into a full circuit model in matrix form, from which we solved the eddy currents induced in the vacuum-vessel walls using the conjugate gradient (CG) method to efficiently handle the large sparse system. Once the eddy currents were obtained, the loop voltage was computed using the Biot–Savart law, considering the contributions from the central solenoid and the eddy currents in the vacuum-vessel walls. The calculation shows that eddy current initially suppresses the loop voltage but decay rapidly, with their influence reducing to below 5% after 0.5 ms. Consequently, the central solenoid current should be maintained longer than 0.5 ms to minimize eddy current effects.

To model the plasma response, we incorporated Spitzer and neoclassical[18] resistivity models. In addition, we employed atomic data from the ADAS[17] to calculate ionization and recombination rate coefficients as functions of electron temperature and density. These physics-based parameters were integrated into a time-dependent plasma evolution solver, which computes the time evolution profiles of plasma temperature, current, and ionization fraction. This allows us to evaluate whether a given central solenoid current profile is

sufficient to initiate plasma breakdown and sustain current development under specific initial conditions.

According to our model calculations, achieving breakdown at a particle density of $n = 10^{17}\ m^{-3}$ (corresponding to a pressure of 10^{-5} Torr) with a connection length of 500 m, a loop voltage of approximately 0.046 V is required. Under these conditions, the central solenoid current change rate must exceed 289 A/ms. Furthermore, to achieve a plasma temperature of 10 eV in the mini-Tokamak with a major radius of 85 mm, minor radius of 55 mm, and elongation $\kappa = 1.82$, the central solenoid current should decrease linearly from 5.4 kA to 0 kA in 10 ms, followed by a ramp from 0 kA to –1.5 kA in 40 ms. When the particle losses and the corresponding energy losses are neglected, the plasma temperature can be maintained at 10 eV for 40 ms.

In conclusion, this work provides comprehensive models for analyzing tokamak startup physics. The developed models not only support the design of effective central solenoid current profile but also offer predictive insight into plasma initiation and stability under various operating conditions. These models are applicable to other tokamak devices and may serve as a reference for future experimental validation and hardware optimization. As an example, the model has been applied to FIRST, with detailed calculations presented in Appendix A.15.

Table 5: Parameters and preferable initial condition for mini-Tokamak.

Parameters	mini − Tokamak
R	85 mm
a	55 mm
κ	1.82
P_0	10^{-5} Torr
n_0	10^{17} m^{-3}
T_0	1 eV
$I_{\mathrm{cs\_max,1}}$	5.4 kA
$\Delta t_{\mathrm{cs\_1}}$	10 ms
$I_{\mathrm{cs\_max,2}}$	−1.5 kA
$\Delta t_{\mathrm{cs\_2}}$	40 ms
$V_{\mathrm{loop\_1}}$	0.086 V
$V_{\mathrm{loop\_2}}$	0.006 V
$I_{\mathrm{p,max}}$	9.7 A
$T_{\mathrm{p,max}}$	10.7 eV

Reference

[1] Sarang. Proton–Proton Chain Reaction, Wikimedia Commons, 2016. *https://en.wikipedia.org/wiki/Nuclear_fusion#/media/File:Fusion_in_the_Sun.svg*.

[2] Wykis. Deuterium–tritium Fusion Reaction [Image], Wikimedia Commons, 2007. *https://commons.wikimedia.org/wiki/File:Deuterium-tritium_fusion.svg*.

[3] ChemConnections. Stages of the Inertial Confinement Fusion Process [Image]. *https://chemconnections.org/crystals/icf.html.*

[4] U.S. Department of Energy. Magnetic field structure in a tokamak [Image]. *https://www.energy.gov/science/doe-explainstokamaks.*

[5] Zhe Gao. Comparison of spherical tokamak to standard tokamak [Image], *Matter and Radiation at Extremes*, 1(3):153–162, June 2016. Fig. 1(a).

[6] Po-Yu Chang. 國科會磁約束高溫電漿計畫-子計畫一執行情形.

[7] H.-T. Kim *et al*. Development of full electromagnetic plasma burn-through model and validation in MAST. *Nuclear Fusion*, 62, 126012 (2022).

[8] Fluid Mechanics 101. Conjugate Gradient [Video]. YouTube, 2023. *https://www.youtube.com/watch?v=MdPhVsgTc1Q*.

[9] O. Alexandrov. Conjugate gradient path [Image], Wikimedia Commons, 2007. *https://en.wikipedia.org/wiki/Conjugate_gradient_method#/media/File:Conjugate_gradient_illustration.svg*.

[10] F. W. Grover, *Inductance Calculations: Working Formulas and Tables*. New York: D. Van Nostrand Company, 1946.

[11] MathWorks. Preconditioned Conjugate Gradient (pcg) – Solve system of linear equations. *https://www.mathworks.com/help/matlab/ref/pcg.html.*

[12] OpenStax. Solenoid self-inductance. Electricity and Magnetism – Section 14.3: Self-Inductance and Inductors. *https://phys.libretexts.org.*

[13] Dougsim. Visualization of Townsend Avalanche [Image]. Wikimedia Commons, 2012. *https://commons.wikimedia.org/wiki/File:Electron_avalanche.gif*.

[14] A Fridman, A Chirokov and A Gutsol. Non-thermal atmospheric pressure discharges. *Journal of Physics D: Applied Physics*, 38, January 2005.

[15] H.-T. Kim *et al*. Enhancement of plasma burn-through simulation and validation in JET. *Nuclear Fusion*, 52, 2012.

[16] F. Trintchouk *et al.* Measurement of the transverse Spitzer resistivity during collisional magnetic reconnection. *Physics of Plasmas*, 10, 2003.

[17] ADAS – Atomic Data and Analysis Structure, *ADAS*. *https://open.adas.ac.uk/*.

[18] NEOS Development Team. NEOS: Open-source neoclassical transport code. *https://gitlab.epfl.ch/spc/public/NEOS*.

[19] FreeGS. Free-boundary Grad–Shafranov solver for plasma equilibrium reconstruction. *https://github.com/freegs-plasma/freegs*.

[20] COMSOL. Simulate real-world designs, devices, and processes with multiphysics software from comsol. *https://www.comsol.com/*.

A Appendix

A.1 Gyro radius calculation

In a magnetically confined plasma, charged particles undergo gyro-motion around magnetic field lines. The gyro-radius depends on the particle species, temperature, and local magnetic field strength. For effective confinement, the gyro-radius must be significantly smaller than the plasma minor radius, particularly near the plasma edge, to avoid particle loss to the vacuum-vessel wall.

In our mini-Tokamak design, the magnetic field at the plasma center (R = 85 mm) is 0.1 T. Based on the inverse radial scaling of toroidal field strength $B \propto 1/R$, we estimate the field to be 0.283 T at the inner edge (R = 30 mm) and 0.061 T at the outer edge (R = 140 mm), as shown in Figure 49. The corresponding distances from the inner and outer plasma boundaries to the vacuum-vessel walls are approximately 11 mm and 17 mm, respectively.

To evaluate whether different species can be confined under these conditions, we calculate the gyro-radius for electrons, H^+, He^+, and Ar^+ ions at three representative positions of the plasma: the inner side (R = 30 mm), center (R = 85 mm), and outer side (R = 140 mm),

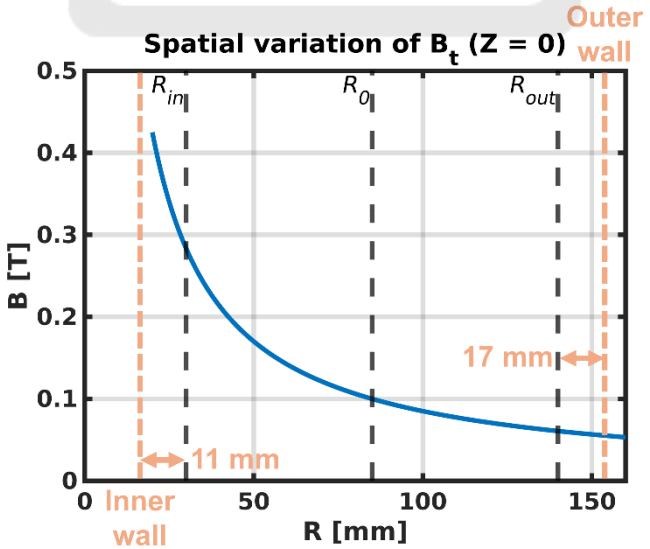

Figure 49: Spatial variation of toroidal magnetic field in the midplane (z = 0)

for temperatures ranging from 1 eV to 30 eV. The gyro radius of a charged particle is given by the following expression:

$$r_L = \frac{\sqrt{2mk_B T}}{qB} \tag{121}$$

where m is the particle mass in kg, q is the particle charge, B is the magnetic field strength in T, T is the particle temperature in eV, and $k_B = 1.602 \times 10^{-19}$ J/eV is the Boltzmann constant. The corresponding results are shown in Figure 50(a) to (c).

From these results, we observe that electrons remain well confined at all radial positions up to a temperature of 30 eV, with their gyro-radii significantly smaller than the plasma

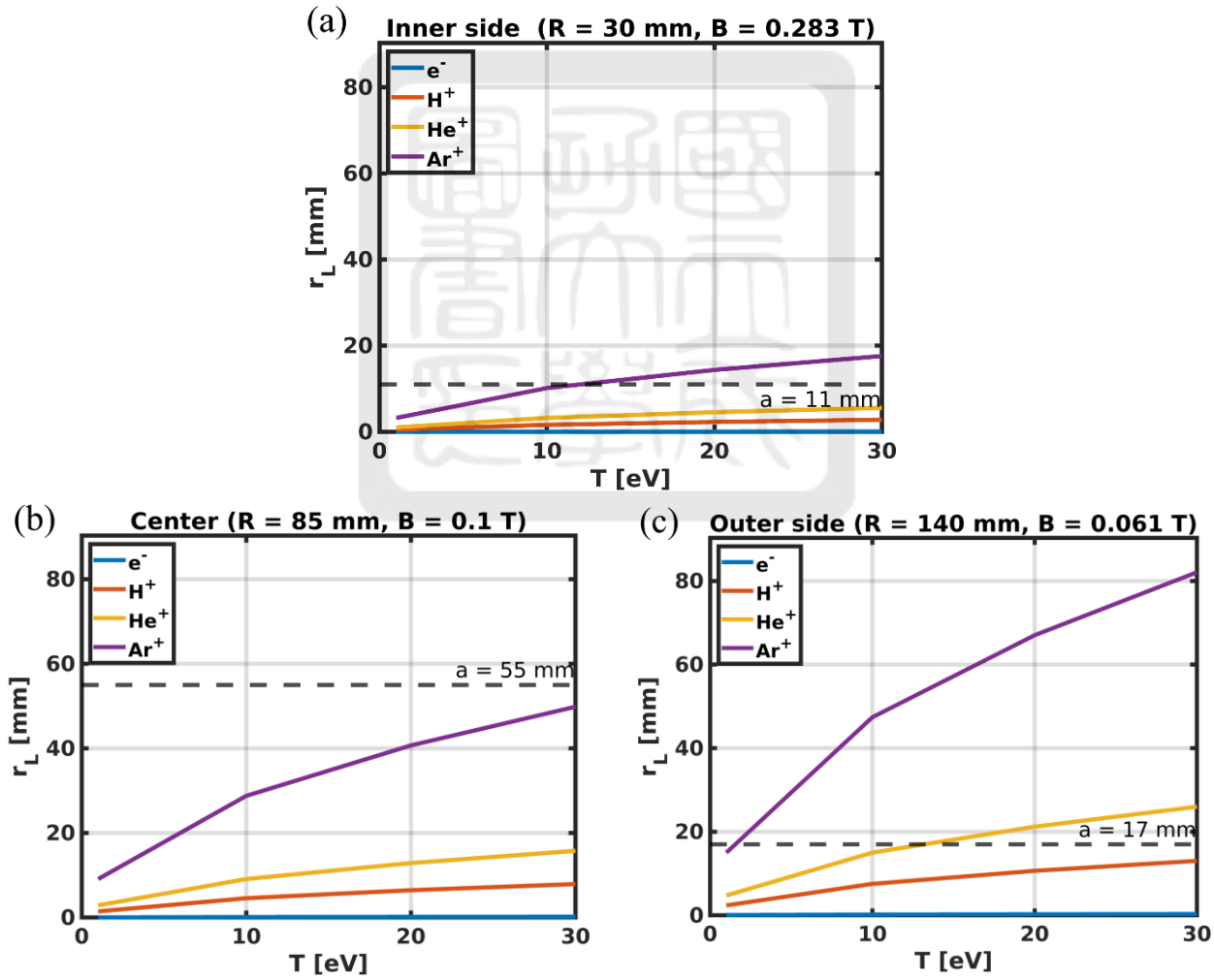

Figure 50: (a) Gyro-radius at plasma inner edge. (b) Gyro-radius at plasma center. (c) Gyro-radius at plasma outer edge.

minor radius. H^+ ions also remain well confined at all radial positions up to a temperature of 30 eV, with their gyro-radii smaller than the plasma minor radius. He^+ ions are well confined at lower temperatures, but at the outer edge, as shown in Figure 50(c), their gyro-radius exceeds the 17 mm confinement margin when the temperature exceeds 13 eV. In contrast, Ar^+ ions exhibit substantially larger gyro-radii throughout the plasma. At the inner edge, as shown in Figure 50(a), Ar^+ ions exceed the 11 mm confinement limit above 10 eV, and at the outer edge, they exceed the 17 mm limit even at 2 eV.

Based on these findings, both hydrogen and helium can be used as working gas. However, due to safety, helium was selected as the working gas for the mini-Tokamak. While argon is commonly used for diagnostics in larger machines, it cannot be effectively confined in this device. Additionally, to ensure adequate magnetic confinement of helium ions, the operational plasma temperature is therefore limited to 10 eV.

A.2 mini-Tokamak equilibrium

The goal of this section is to verify whether the plasma in the mini-Tokamak can be magnetically confined under the given hardware constraints. In particular, we aim to determine the required currents in the poloidal field coils (PFCs) that are needed to achieve equilibrium. Notice that an equilibrium state will be designed but stabilization check is beyond the scope of this thesis. This appendix is divided into three parts. Appendix A.2.1 introduces the theoretical foundation of plasma equilibrium in tokamak, starting from the basic magnetohydrodynamic (MHD) force balance and leading to the Grad–Shafranov equation. Then, Appendix A.2.2 presents the equilibrium calculation results obtained using the Grad–Shafranov solver FreeGS[19], applied to the mini-Tokamak configuration. Last, Appendix A.2.3 investigates how variations in the plasma current spatial profile affect the equilibrium and safety factor distribution.

A.2.1 Equilibrium theory

To maintain an equilibrium plasma configuration in a tokamak, the forces acting on the plasma must be in balance. The fundamental condition for this is the MHD force balance equation:

$$\vec{J} \times \vec{B} = \nabla p \tag{122}$$

where $\vec{J}$ is the current density, $\vec{B}$ is the magnetic field, and p is the plasma pressure. This equation means that the magnetic force acting on the plasma, also known as the Lorentz force, must be balanced by the internal pressure gradient. In other words, equilibrium is achieved when the inward magnetic forces are exactly offset by the outward pressure forces.

In axisymmetric configurations such as tokamak or spherical tokamak, the MHD equilibrium condition can be reformulated into a single partial differential equation known as the Grad–Shafranov equation. This equation determines the structure of magnetic flux surfaces and the distribution of plasma current in the poloidal cross-section of the device. Its

derivation involves decomposing the magnetic field into toroidal and poloidal components, and applying the assumption of axisymmetry, i.e., $\partial/\partial\phi = 0$.

The Grad–Shafranov equation under axisymmetric assumptions can be expressed as:

$$\frac{\partial^2\psi}{\partial r^2} - \frac{1}{r}\frac{\partial\psi}{\partial r} + \frac{\partial^2\psi}{\partial z^2} = -\mu_0 r^2 \frac{dp}{d\psi} - \frac{1}{2}\frac{dF^2}{d\psi} \tag{123}$$

where $\psi(r,z)$ is the poloidal magnetic flux, which describes the shape of magnetic surfaces in the cross-section; $p(\psi)$ is the plasma pressure, assumed to vary along magnetic flux surfaces; $F(\psi) = rB_\phi$ represents the toroidal magnetic field contribution, with B_ϕ being the toroidal magnetic field component, and μ_0 is the vacuum permeability. This equation describes how the electromagnetic force balances the pressure gradient to determine the equilibrium structure of the plasma. The left-hand side represents the spatial variation of the poloidal magnetic flux in cylindrical coordinate. The right-hand side contains the source terms due to plasma pressure and toroidal field contribution.

By solving the Grad–Shafranov equation under specified boundary conditions and prescribed plasma current and pressure profile, and the required plasma shape, one can calculate the required coil current to maintain equilibrium. These equilibrium calculations and results will be presented in the next section.

A.2.2 Equilibrium calculation

Based on the theoretical framework described in the previous section, equilibrium calculations were performed using the FreeGS code[19]. The objectives of these calculations were to evaluate whether the designed mini-Tokamak could achieve the desired plasma shape, as defined by the aspect ratio and elongation specified in Chapter 2, and to determine the required currents in the PFCs (P1U, P1L, P2U, and P2L) necessary to maintain equilibrium with a desired plasma current. These PFCs currents are driven by pulse-width modulation (PWM) in the mini-Tokamak, with further details provided in Appendix A.3. These calculations were conducted under the constraints of the mini-Tokamak's hardware

configuration, including the geometry of the vacuum vessel, the location of the central solenoid, and the locations of the poloidal field coils, as described in Chapter 2.

The spatial arrangement of the PFCs used in the FreeGS is illustrated in Figure 51. Four coils were employed:

- P1U and P1L are located at $(r = 0.20\ \mathrm{m}, \mathrm{z} = 0.10\ \mathrm{m})$ and $(r = 0.20\ \mathrm{m}, \mathrm{z} = -0.10\ \mathrm{m})$, respectively. These coils are positioned symmetrically above and below the midplane near the outer edge of the vacuum vessel. They primarily serve to control the vertical shape of the plasma and help maintain the equilibrium.
- P2U and P2L are placed inside the vacuum vessel and closed to the plasma, at $(r = 0.10\ \mathrm{m}, \mathrm{z} = 0.15\ \mathrm{m})$ and $(r = 0.10\ \mathrm{m}, \mathrm{z} = -0.15\ \mathrm{m})$, respectively. These inner coils are used to adjust the plasma shape near the edge, particularly influencing the location of the separatrix and the structure of the magnetic flux surfaces.

All four coils were treated as independently controlled in the calculations, allowing their currents to be automatically adjusted by the equilibrium solver according to a specified isoflux constraint. This constraint defines the plasma boundary as a closed contour of

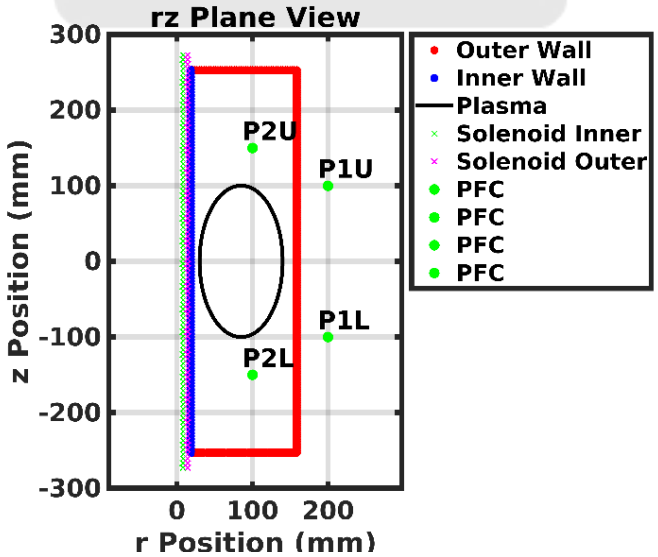

Figure 51: Coil and chamber layout of the mini-Tokamak (rz plane).

constant poloidal magnetic flux (ψ), constructed from four connected line segments that enclose the desired separatrix, the blue dashed line in Figure 52. The calculation was configured for a plasma current of 10 kA and a central pressure of 0.32 Pa, i.e., $n = 7.7 \times 10^{-19}\ \mathrm{m}^{-3}$ for $T = 10$ eV, and a vacuum toroidal field strength characterized by $f = RB_{\mathrm{t}} = 8.5 \times 10^{-3}$ Tm.

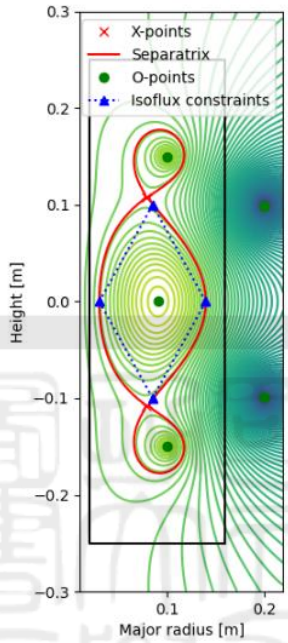

Figure 52: Equilibrium contours and isoflux constraints in FreeGS.

To define the current distribution within the plasma, a parametric profile for the toroidal plasma current density J_ϕ was used, expressed as a function of the normalized poloidal flux ψ_{n}:

$$J_\phi \propto \left(1 - \psi_{\mathrm{n}}^{\alpha_{\mathrm{m}}}\right)^{\alpha_{\mathrm{n}}} \tag{124}$$

where $\psi_{\mathrm{n}} = 0$ at the magnetic axis and $\psi_{\mathrm{n}} = 1$ at the plasma edge. In this calculation, $\alpha_{\mathrm{m}} = 2$ and $\alpha_{\mathrm{n}} = 2$ were selected, resulting in a moderately broad current profile, as shown in Figure 53(a). This shape provides a compromise between core peaking and profile flatness, offering favorable stability and confinement properties, which will be detailed describe in the next section.

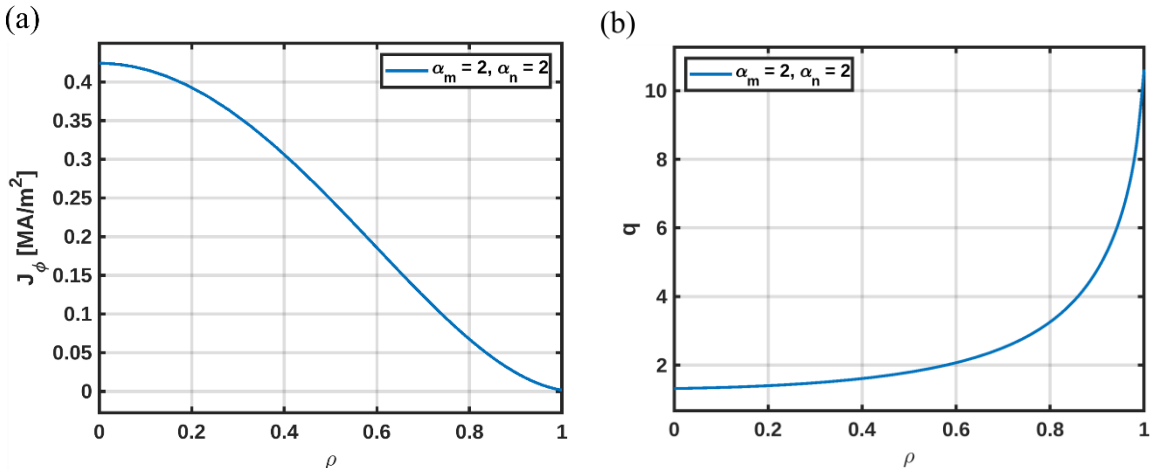

Figure 53: (a) Toroidal current density profile in FreeGS. (b) Safety factor profile in FreeGS.

The resulting equilibrium is illustrated in Figure 52. The corresponding PFCs currents were shown in Table 6. These coil currents generated a magnetic configuration capable of achieving a well-confined plasma, with a safety factor profile ranging from $q_{\min} = 1.39$ at the core to $q_{95} = 6.32$ at the edge, as shown in Figure 53(b). The computed poloidal beta was $\beta_p = 2.49 \times 10^{-4}$, and the magnetic axis remained centered near the designed plasma major radius. The results confirmed that the initial PFC positioning and symmetry are sufficient for achieving equilibrium in the mini-Tokamak configuration. This validated magnetic structure and corresponding coil currents provide a solid foundation for future experimental implementation.

Table 6: Currents in the poloidal field coils.

Coil	Current (A)
P1U	−4295.6
P1L	−4295.8
P2U	2252.8
P2L	2253.1

A.2.3 Effect of α_{m} and α_{n} on current and safety factor profile

In the FreeGS equilibrium calculation, the toroidal current density profile J_{ϕ} is prescribed as a function of the normalized poloidal flux ψ_n, defined such that $\psi_{\mathrm{n}} = 0$ at the plasma center and $\psi_n = 1$ at the edge. The profile takes the form:

$$J_{\phi} \propto \left(1 - \psi_{\mathrm{n}}^{\alpha_{\mathrm{m}}}\right)^{\alpha_{\mathrm{n}}} . \tag{125}$$

Here, α_{m} and α_{n} are shaping parameters that control how the current is distributed across the plasma cross-section. The parameter α_{m} controls the broadness of the profile by modifying the inner exponent of ψ_{n}, which determines how gradually the current density decreases from the center toward the edge. A larger α_{m} results in a flatter and broader current profile. Figure 54(a) shows as α_{m} increases from 2 to 4 (with α_{n} held constant), corresponding to the blue, red, and yellow line respectively, the profile transitions from a steeply declining shape to a much broader and flatter distribution. Specifically, when α_{m} is small (e.g., $\alpha_{\mathrm{m}} = 2$), the current is strongly concentrated near the plasma center and rapidly drops off toward the edge. As α_{m} increases, the central plateau becomes wider, and the gradient near the edge becomes more gradual. This demonstrates how α_{m} modulates the "core-flatness" of the profile, controlling the width of the region with high current density.

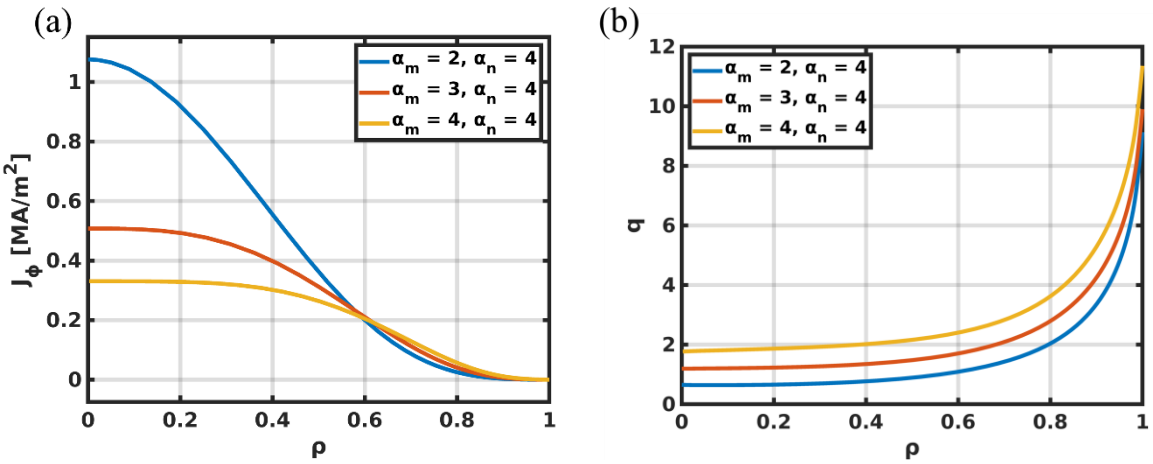

Figure 54: (a) Toroidal current density profile in FreeGS. (b) Safety factor profile in FreeGS.

In contrast, α_n controls the peakness of the profile. As the outer exponent, it governs how sharply the current density falls off once ψ_n approaches 1. A larger α_n increases the central current density and produces a more peaked distribution. As shown in Figure 55, as α_n increases from 2 to 4 (with α_m held constant), the current density profile becomes increasingly peaked near the magnetic axis ($\psi_n = 0$), while the edge current density drops more steeply. This demonstrates that α_n effectively sharpens the profile, concentrating more current toward the center and suppressing the edge contribution.

As illustrated in Figure 54(a) and (b), increasing α_m leads to a broader distribution and reduces the peak current near the center. This tends to raise the core safety factor q_{min}, which helps suppress tearing modes. However, overly broad profiles can reduce magnetic shear in the core, which may weaken internal stability or degrade confinement.

On the other hand, as shown in Figure 55(a) and (b), increasing α_n makes the current profile more peaked, lowering the central q-value while steepening the safety factor gradient near the edge. This enhances magnetic shear at the boundary, which is favorable for stabilizing edge-localized modes and supporting the formation of transport barriers.

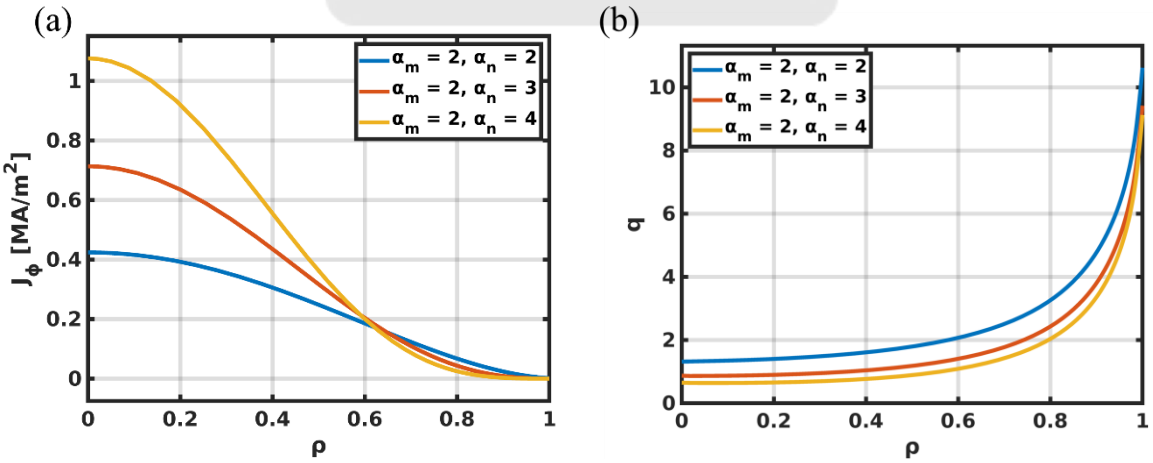

Figure 55: (a) Toroidal current density profile in FreeGS. (b) Safety factor profile in FreeGS.

Nonetheless, excessive edge shear may also alter the overall MHD stability balance and trigger other instabilities such as peeling-ballooning modes.

In summary, the choice of α_m and α_n shapes the current profile and directly affects the safety factor distribution. These parameters must be carefully tuned to balance core and edge stability and to achieve the desired confinement performance in the mini-Tokamak. However, due to hardware limitations, the current profile in the mini-Tokamak cannot be freely adjusted in practice. Therefore, the present calculations serve primarily as a reference, and the final configuration must be determined based on experimental results.

A.3 Magnetic ripples from PWM signal

In the mini-Tokamak, the poloidal field coils (PFCs) are driven using a Pulse-Width Modulation (PWM), which generates magnetic ripples that might penetrate the vacuum-vessel wall and alter the internal magnetic field configuration. Therefore, it is necessary to ensure that these ripples do not adversely affect the experiment.

This appendix consists of two parts. Appendix A.3.1 introduces PWM and presents the magnetic ripple calculation obtained using our model and COMSOL simulations. Appendix A.3.2 provides the conclusion.

A.3.1 PWM-induced magnetic ripples

From Chapter 5, we now have the central solenoid current profile and the evolution of plasma parameters. Assuming successful plasma initiation, plasma reaches the desired equilibrium state, as shown in Figure 52, which illustrates the equilibrium shape calculated using FreeGS[19]. In this configuration, the central solenoid is responsible for gas breakdown and driving the plasma current. The toroidal field coils generate a strong toroidal magnetic field, which causes charged particles to gyrate along the magnetic field lines in the toroidal direction, as indicated by the light blue arrows in Figure 24. In addition, PFCs are required to produce vertical magnetic fields (B_Z), shaping the overall magnetic configuration inside the tokamak. These vertical fields are essential for achieving a rational magnetic field distribution, leading to the desired equilibrium shape shown in Figure 52. The required PFCs currents that generate this field structure have been computed using FreeGS and are listed in Table 6 in Appendix A.2.2.

In our system, the currents in the PFCs are supplied using PWM, which operates by rapidly switching the applied voltage on and off at a fixed frequency. By varying the duty cycle, the fraction of time that the voltage is “ON” during each cycle varies. Therefore, the average voltage, and hence the coil current, can be controlled precisely.

To study the effect on the PWM signal, we provided a simulated current profile with PWM signal. The current is provided by the circuit shown in Figure 56. When SW_1 is ON and SW_2 is OFF, the circuit can be obtained by solving:

$$V_0 - IR - L\frac{dI}{dt} = 0. \tag{126}$$

When SW_1 is OFF and SW_2 is ON, the circuit can be obtained by solving:

$$-IR - L\frac{dI}{dt} = 0. \tag{127}$$

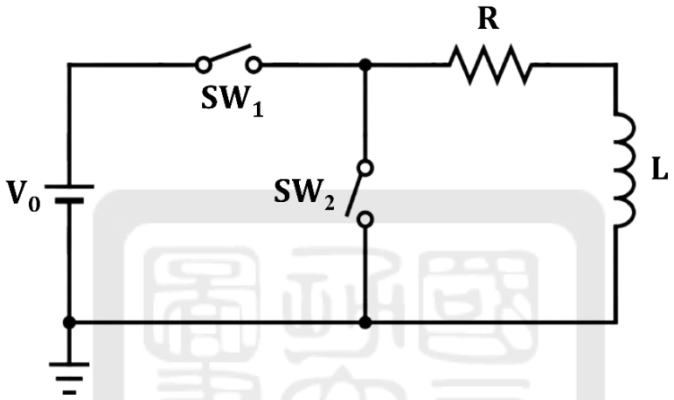

Figure 56: PWM circuit.

In both equations, L is the coil's self-inductance and R is the coil resistance. When voltage is applied (SW_1 is ON, SW_2 is OFF), the current ramps up; when the voltage is turned off (SW_1 is OFF, SW_2 is ON), the current ramps down, resulting in periodic ripples. These current ripples can generate fluctuating magnetic fields that may penetrate through the vacuum-vessel wall and potentially affect the internal magnetic field distribution.

Figure 57 illustrates the typical behavior of a coil driven by PWM. The orange dashed line shows the applied voltage across the PFC, which alternates between 0 V and 20 V in a square-wave pattern. The resistance and inductance here is 0.02 Ω and 10 μH. This switching occurs at a fixed frequency 1 kHz and duty cycle, characteristic of PWM control. The blue solid line represents the resulting current through the coil. Due to the coil's inductance, the current cannot follow the abrupt voltage changes instantaneously. Instead, it

increases as one minus an exponential function during each “ON” phase of the voltage (when the voltage is high) and then decreases exponentially during the “OFF” phase (when the voltage drops to zero). This produces the characteristic sawtooth-shaped current waveform.

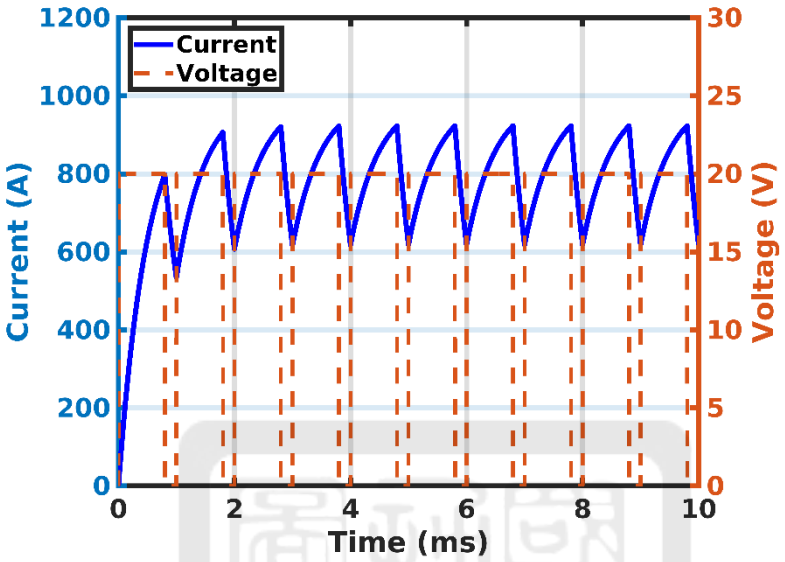

Figure 57: PWM voltage and current waveforms for 1 kHz.

To mitigate this issue, it is essential to evaluate the frequency range at which the vacuum-vessel wall can effectively shield out these ripple-induced magnetic fields. This shielding effect arises due to the eddy current generated in the conductive vacuum-vessel wall. The induced eddy current in the conducting wall generate opposing magnetic fields, which cancel out the penetrating magnetic field components. Until the eddy current damp to zero by the wall resistivity, the field penetrates through the vacuum-vessel wall. For high frequency ripples, the eddy current doesn’t have enough time to damp out. As a result, the high frequency ripples are cancelled by the eddy current and thus shielded out. The damping time strongly depends on the resistance of the eddy current path. In other words, it depends on the cross section of the eddy current path. The skin depth δ, which determines how far a magnetic field can penetrate into a conductor, is given by:

$$\delta = \sqrt{\frac{2}{\mu\sigma\omega}} \tag{128}$$

where μ is the magnetic permeability, σ is the electrical conductivity of the wall material, and $\omega = 2\pi f$ is the angular frequency of the PWM ripple. When the skin depth becomes significantly smaller than the wall thickness, the PWM-generated magnetic field is strongly cancelled and cannot penetrate the interior of the vacuum vessel. Therefore, by ensuring the PWM frequency is sufficiently high, the ripple fields can be effectively shielded by the vacuum-vessel wall, minimizing their impact on the magnetic field.

In addition, as the frequency increases, the amplitude of the PWM ripples in current also decreases. This is because at higher switching frequencies, each ON-OFF cycle becomes shorter, leaving less time for the current to rise or fall significantly within each period. As a result, the ripple magnitude is reduced, as shown in Figure 58. Increasing the frequency not only enhances the vacuum-vessel wall's attenuation of the magnetic ripple, but also intrinsically reduces the ripple amplitude. Therefore, we identify an appropriate high-frequency range that ensures both effective shielding and minimal ripple-induced disturbance.

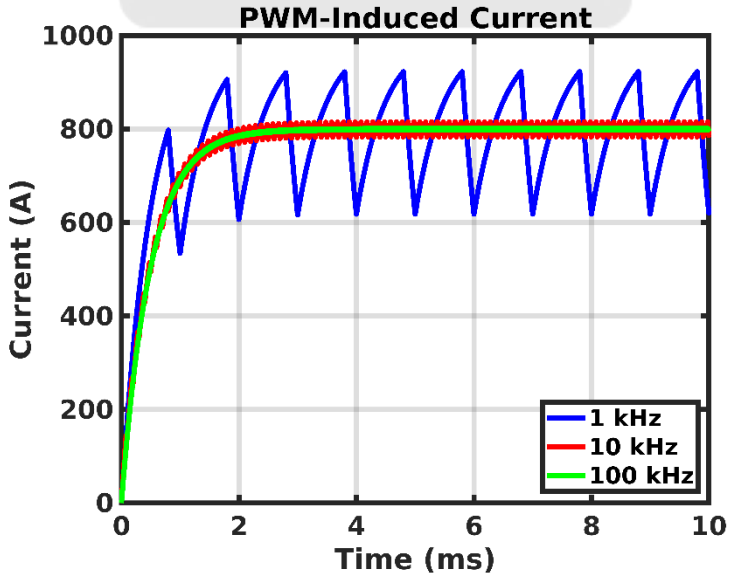

Figure 58: Simulated PWM-induced current waveforms at 1 kHz, 10 kHz, and 100 kHz.

To study the shielding effect of the vacuum-vessel wall, we use a single-turn PFC with a rectangular cross-section of 5 mm × 5 mm to generate a magnetic field with ripples. The corresponding resistance and inductance of the PFC is 0.02 Ω and 10 μH, respectively. Then the coil current with ripples can be provided using the model in Figure 56. Finally, we can calculate the magnetic field in the interior of the vacuum vessel with and without the eddy current of the vacuum-vessel wall for comparison.

(1) COMSOL simulation

We used the "Time-dependent", "Magnetic Fields (mf)", and "2D axisymmetric" modules in COMSOL[20] to simulate the magnetic field distribution generated by the PWM current in the PFC within the interior of the vacuum vessel. First, we constructed the geometry of the mini-Tokamak based on the specifications described in Chapter 2, along with a test PFC defined as a single-turn circular loop with a rectangular cross-section of 5 mm × 5 mm, located at a major radius of 170 mm and z = 0, as illustrated by the small dot in Figure 59. A semicircular outer shell was added and defined as an “infinite element domain” to prevent artificial boundary effects on the simulation, as shown in Figure 59.

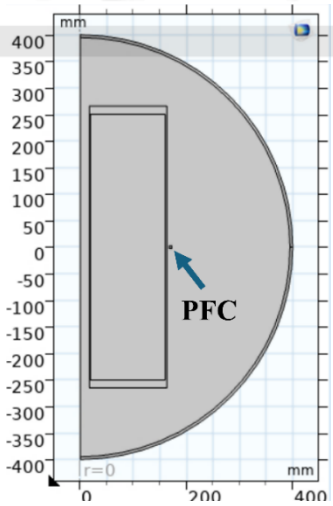

Figure 59: mini-Tokamak configuration in COMSOL.

The vacuum vessel material was set to 304L stainless steel, the PFC was assigned to copper, and all other regions, including the interior of the vacuum vessel and the infinite shell, were set as air. The PFC was defined as a "Coil" domain in COMSOL, and the PWM current profile (blue solid line in Figure 57) was imported as the time-dependent current source for the PFC.

To calculate the response of the vacuum-vessel wall to the time-varying magnetic field, "Ampère's Law in solids" was applied to the vacuum vessel region. This setting enables the simulation of eddy current induced in conductive materials by time-varying magnetic fields. Unlike "Ampère's Law" for non-conductive regions (which only solves for magnetic fields), the "in solids" variant includes Ohm's law to account for induced current, making it essential for capturing the electromagnetic shielding effects of the vacuum-vessel wall.

Figure 60 shows the magnetic field distribution simulated using COMSOL under different PWM frequencies. In each subplot, the red line represents the magnetic field generated solely by the PWM current in the PFC, while the blue line includes the contribution from both the PFC and the eddy current induced in the vacuum-vessel wall. At 1 kHz, as shown in Figure 60(a), the magnetic field exhibits a pronounced ripple pattern with visible peak-to-peak oscillations, indicating that the vacuum-vessel wall cannot fully cancel the low-frequency components. However, the blue line still shows noticeable damping compared to the red one, demonstrating the partial shielding effect of the vessel.

As the frequency increases to 10 kHz in Figure 60(b), the ripple amplitude visibly decreases, and the magnetic field becomes smoother, although some residual oscillation remains. By 100 kHz, shown in Figure 60(c), the ripple is almost non-noticeable in the blue curve, suggesting that no significant high-frequency components occurs in the vacuum vessel.

To further quantify the attenuation of magnetic ripple caused by the vacuum-vessel wall at different PWM frequencies, Figure 61 presents the extracted ripple components for the same three cases shown in Figure 60. The ripple amplitude is obtained by subtracting the moving average of each magnetic field signal from its original waveform. The number of moving-averaged points for 1-kHz case, 10-kHz case, and 100-kHz case are all 50 points. In Figure 61(a), corresponding to the 1 kHz case, both the "with chamber" and "no chamber" curves show clear periodic oscillations. However, the blue curve (with chamber) has visibly reduced amplitude. The peak ripple reaches approximately $\pm 2 \times 10^{-4}$ T in the presence of

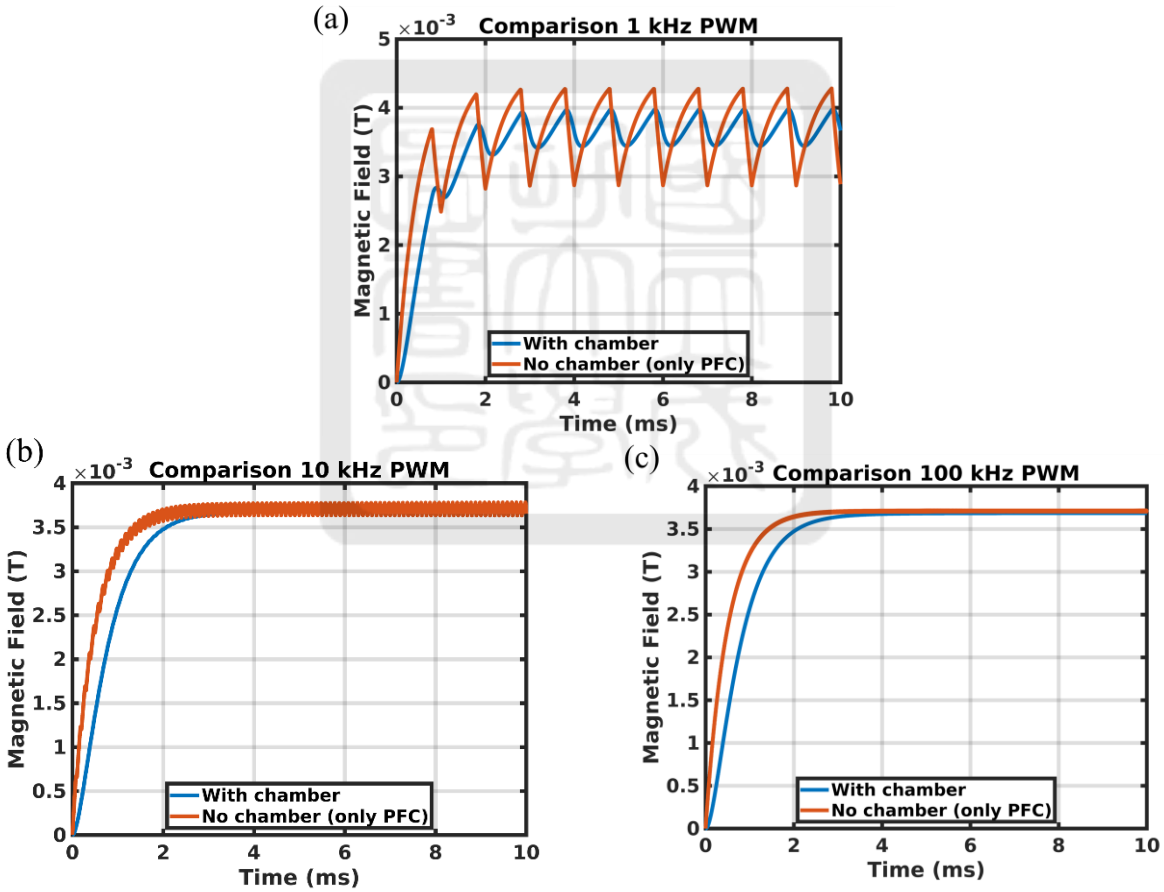

Figure 60: (a) Magnetic field comparison at 1 kHz PWM with and without chamber.

(b): Magnetic field comparison at 10 kHz PWM with and without chamber.

(c): Magnetic field comparison at 100 kHz PWM with and without chamber.

the chamber, whereas the no-chamber case (red curve) exceeds $\pm 5 \times 10^{-4}$ T. This shows that while attenuation is present, the ripple still penetrates significantly at 1 kHz.

As the PWM frequency increases to 10 kHz in Figure 61(b), the suppression becomes much more effective. The blue curve's ripple amplitude drops to within $\pm 5 \times 10^{-6}$ T, while the red curve still exhibits strong oscillations exceeding $\pm 7 \times 10^{-5}$ T. At 100 kHz, as shown in Figure 61(c), the difference becomes dramatic. The ripple in the chamber-present case is nearly flat, remaining within $\pm 1 \times 10^{-7}$ T, demonstrating that the eddy current in the vacuum-vessel wall effectively eliminate high-frequency ripple components.

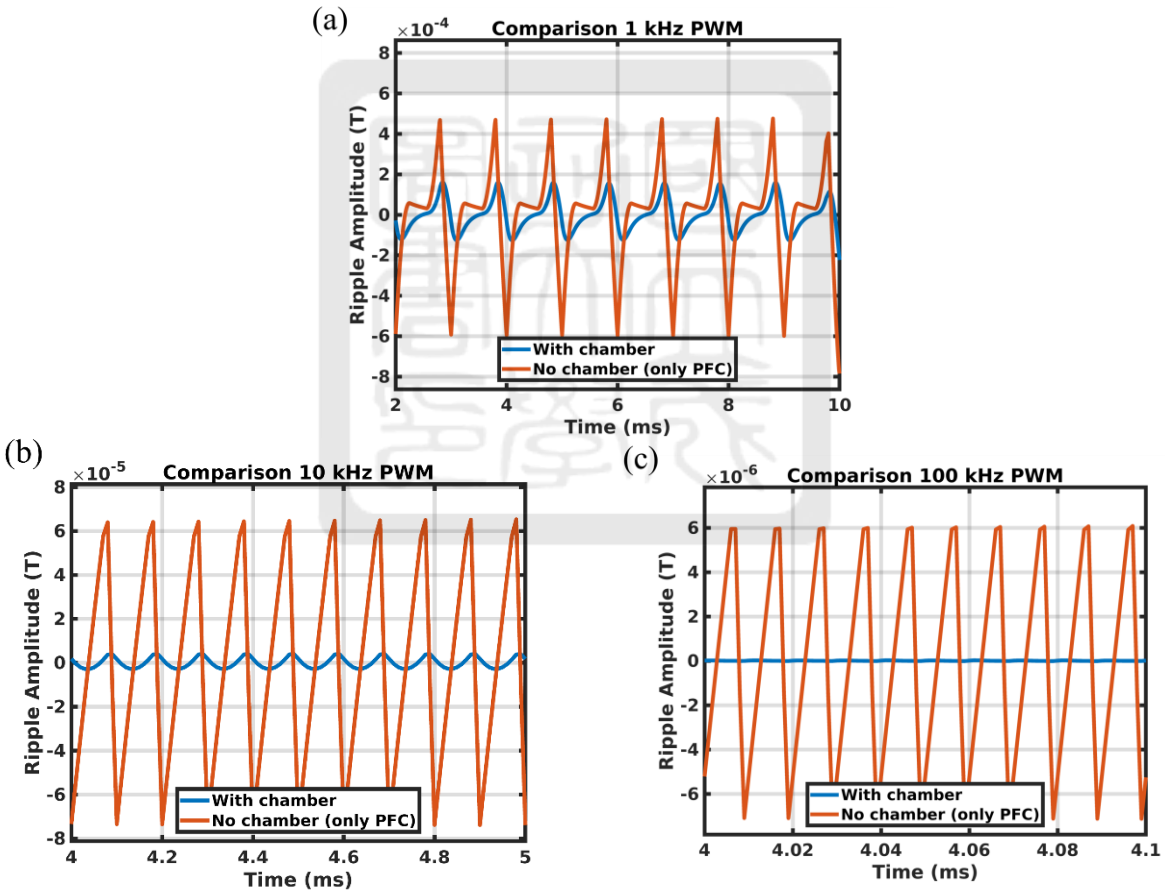

Figure 61: (a) Ripple amplitude at 1 kHz PWM with and without chamber. (b): Ripple amplitude at 10 kHz PWM with and without chamber. (c): Ripple amplitude at 100 kHz PWM with and without chamber.

These results provide strong evidence that increasing the PWM frequency not only reduces the initial amplitude of the magnetic ripple but also significantly enhances the vacuum-vessel wall's ability to attenuate it. At higher frequencies, the vacuum-vessel wall, governed by the skin effect, exhibits greater resistance to time-varying magnetic fields. This confines the induced eddy current to a thinner surface layer, thereby generating opposing magnetic fields that effectively cancel out the ripple components. Although we currently lack experimental data to directly assess the impact of magnetic ripples on plasma equilibrium, the present system operates at a PWM frequency of 1 kHz. Future adjustments to the PWM frequency will be considered based on forthcoming experimental observations and their implications for plasma performance.

(2) MATLAB calculation using our model

Since we have already developed a model capable of calculating the eddy current induced in the vacuum-vessel wall by the PWM current, as described in Chapter 3, and a model for computing the magnetic field generated by both the vacuum-vessel wall and the PFC, as detailed in Chapter 4, we now proceed to validate by comparing results obtained from our model with the results obtained from COMSOL.

We input the vacuum-vessel wall configuration identical to that used in the previous COMSOL simulation, namely the mini-Tokamak geometry based on the specifications described in Chapter 2, along with a test PFC modeled as a single-turn circular loop with a rectangular cross-section of 5 mm × 5 mm, positioned at a major radius of 170 mm and z = 0, as illustrated in Figure 62.

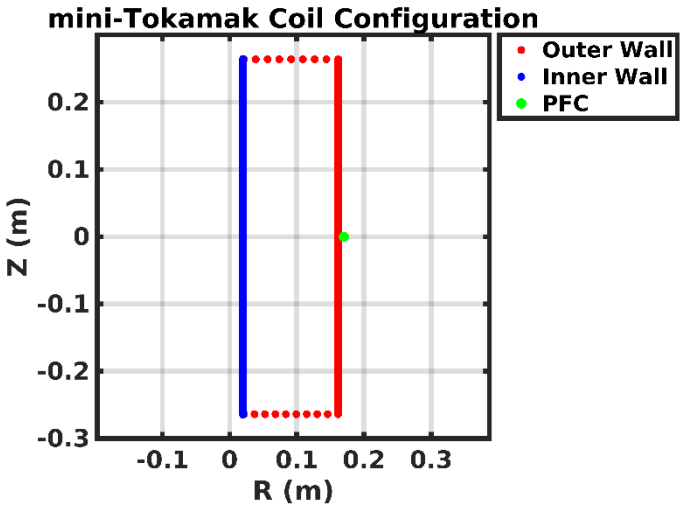

Figure 62: mini-Tokamak coil configuration with a test PFC.

Subsequently, using the same PWM current profile at 1 kHz, 10 kHz, and 100 kHz as described previously, we calculated the eddy current induced in the vacuum-vessel wall. Based on these results, we further computed the magnetic field generated by both the PFC and the induced eddy current, as well as their combined total magnetic field distribution, as shown in Figure 63, which shows the magnetic field B_Z at R = 0.45 m for PWM frequencies of 1 kHz, 10 kHz, and 100 kHz, respectively. For each frequency, we plot the magnetic field generated by the PFC current alone, the field produced by the eddy current in the vacuum-vessel wall, and the combined total field.

As shown in Figure 63(a), at 1 kHz, the vacuum-vessel wall has limited shielding capability, and the eddy current is unable to fully suppress the ripple components from the PFC, resulting in a large oscillatory component in the total field. At 10 kHz (Figure 63(b)), the eddy current begin to effectively oppose the PFC-induced field, reducing the ripple amplitude in the total magnetic field. By 100 kHz, as illustrated in Figure 63(c), the high-frequency ripple is almost unnoticeable, leaving a smooth and nearly steady magnetic field.

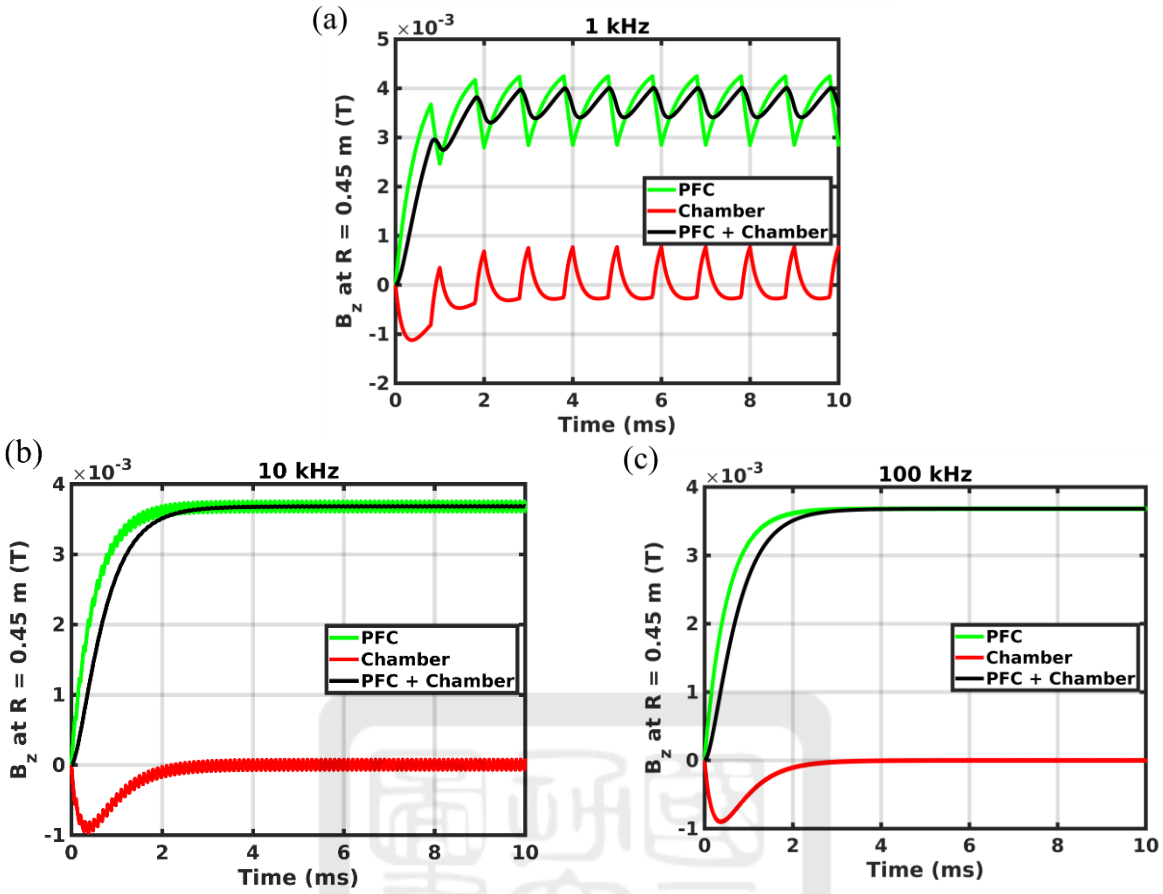

Figure 63: (a) Ripple amplitude at 1 kHz PWM with and without chamber. (b): Ripple amplitude at 10 kHz PWM with and without chamber. (c): Ripple amplitude at 100 kHz PWM with and without chamber.

These results are consistent with the COMSOL simulation findings presented in Appendix A.3.1.1 (Figure 60), confirming that the vacuum-vessel wall's shielding effectiveness improves significantly with increasing PWM frequency. Both our model and the COMSOL simulations exhibit the same trend of ripple suppression, thereby validating the accuracy and reliability of our models.

To further assess the model's accuracy, we performed a direct comparison of the computed total B_Z field from our model with results obtained from COMSOL. As shown in Figure 64, the two results are in excellent agreement across all PWM frequencies, showing that the B_Z waveforms at 1 kHz, 10 kHz, and 100 kHz are nearly identical between the two

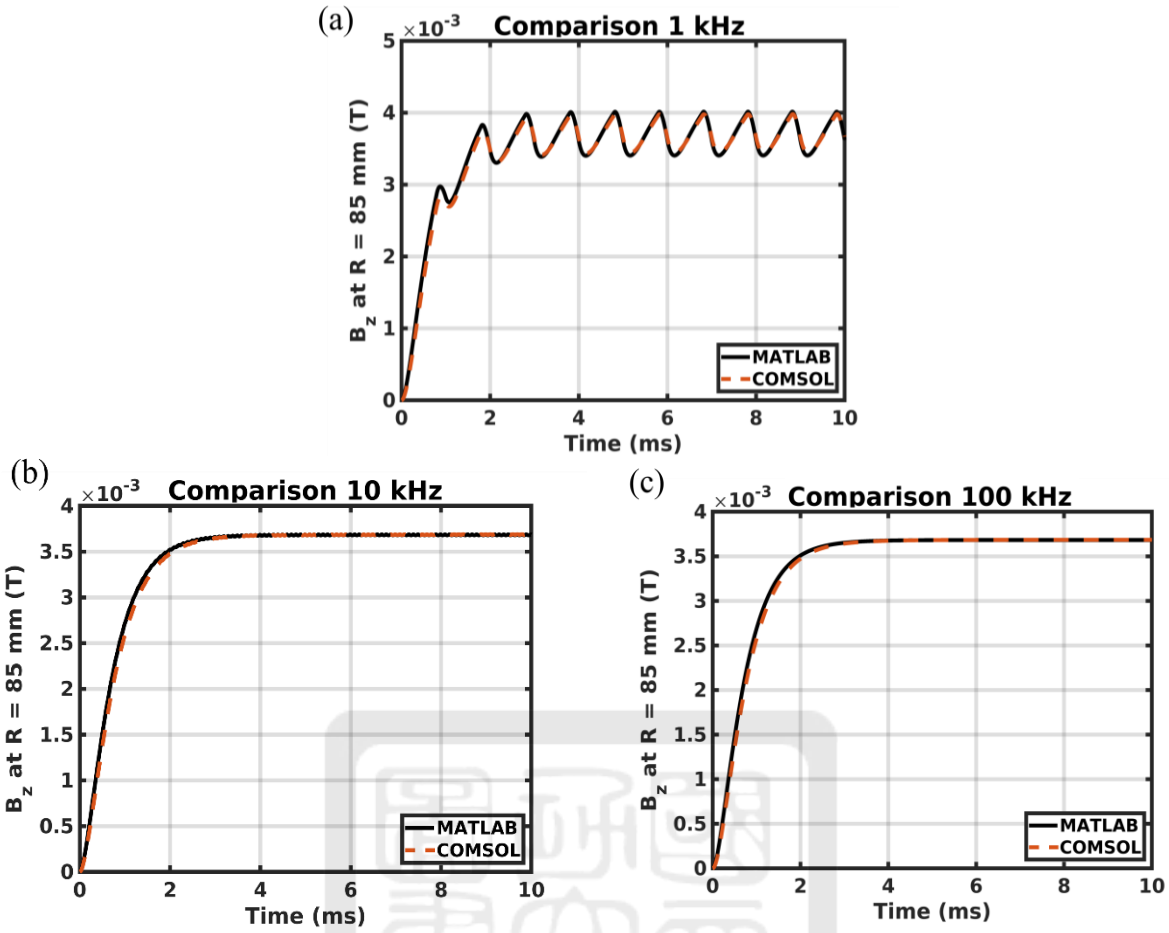

Figure 64: (a) Comparison of B_Z field at 1 kHz PWM frequency. (b): Comparison of B_Z field at 10 kHz PWM frequency. (c): Comparison of B_Z field at 100 kHz PWM frequency.

approaches. It indicates the high accuracy of our models for computing both the eddy current and the magnetic fields.

A.3.2 Conclusion

This appendix confirms that high-frequency PWM ripples is effectively shielded by the vacuum-vessel wall, thereby preventing its penetration into the plasma region; however, the precise frequency for effective shielding should ultimately be determined through experimental validation.

A.4 mini-Tokamak components setup code

```
% The code is in Student_Thesis/2025_ypi/Code/parameter_mini_thesis on the NAS.
% ========== Rec Mode ==========

%% ---- Enable Components ----
enable_chamber = true;
enable_solenoid = true;
enable_plasma = false;
enable_pfc = false;

%% ========== Input Parameters ==========

% Chamber
if enable_chamber
    chamber_height = 531e-3;
    inner_thickness = 1.8e-3;

    inner_distance = 17.25e-3 + inner_thickness/2;
    outer_thk_side   = 3e-3;
    outer_thk_top    = 16e-3;
    outer_thk_bottom = 15e-3;

    outer_x_start = 26.55e-3;
    outer_x_end   = 146.55e-3;
    outer_side_length = outer_x_end - outer_x_start;

    % Inner wall params
    inner_params.height = chamber_height;
    inner_params.thickness = inner_thickness;
    inner_params.distance = inner_distance;

    % Generate inner wall
    [inner_x, inner_z, chamber_components.inner_wall] =
generate_inner_wall(inner_params);

    % Generate outer wall (3 sides only: bottom → left → top)
    num_bottom = 9;
```

```
    z_bot = -chamber_height / 2 + outer_thk_bottom / 2;
    bottom_x = linspace(outer_x_start, outer_x_end, num_bottom)';
    bottom_z = z_bot * ones(num_bottom, 1);

    x_left = 163e-3 - outer_thk_side / 2;
    z_left_start = -chamber_height / 2 + outer_thk_side / 2;
    z_left_end   = chamber_height / 2 - outer_thk_side / 2;
    num_left = ceil((chamber_height) / outer_thk_side);
    left_x = x_left * ones(num_left, 1);
    left_z = linspace(z_left_start, z_left_end, num_left)';

    x_top_start = 152e-3;
    x_top_end   = 24e-3;
    num_top = 9;
    z_top = chamber_height / 2 - outer_thk_top / 2;
    top_x = linspace(x_top_start, x_top_end, num_top)';
    top_z = z_top * ones(num_top, 1);

    outer_x = [bottom_x; left_x; top_x];
    outer_z = [bottom_z; left_z; top_z];
    outer_thk = [ ...
        repmat(outer_thk_bottom, num_bottom, 1); ...
        repmat(outer_thk_side, num_left, 1); ...
        repmat(outer_thk_top, num_top, 1)];
    outer_r = repmat(1.5e-3, length(outer_x), 1);

    chamber_components.outer_wall = struct();
    for i = 1:length(outer_x)
        chamber_components.outer_wall(i).r = outer_x(i);
        chamber_components.outer_wall(i).z = outer_z(i);
        chamber_components.outer_wall(i).thickness = outer_thk(i);
        chamber_components.outer_wall(i).radius = outer_r(i);
        chamber_components.outer_wall(i).coil_length = 2 * pi * outer_x(i);
    end
end

% Solenoid
```

```
if enable_solenoid
    solenoid_radius = 1.3e-3;
    solenoid_radius_cooling = 0e-3;
    isolation_thickness = 1.7e-3;
    solenoid_total_radius = solenoid_radius + isolation_thickness;
    solenoid_inner_num_coils = 92;
    solenoid_layer_count = 2;
    solenoid_outer_num_coils = solenoid_inner_num_coils * (solenoid_layer_count ==
2);
    solenoid_inner_distance = 8e-3;
    solenoid_outer_distance = solenoid_inner_distance + (solenoid_layer_count == 2) * 2
* solenoid_total_radius;
    solenoid_height_between_turns = solenoid_total_radius * 2;
    solenoid_height = solenoid_total_radius * 2 * solenoid_inner_num_coils;
    solenoid_coil_A = pi * (solenoid_radius^2 - solenoid_radius_cooling^2);
    solenoid_a = pi * (solenoid_inner_distance - solenoid_radius)^2;
end

% Plasma
if enable_plasma
    T_plasma = 1; % eV
    n_total_plasma = 1e17; % m^-3
    R_plasma = 85e-3;
    a_plasma = 55e-3;
    kappa = 1.82;
    b_plasma = a_plasma * kappa;
    A_plasma = pi * a_plasma * b_plasma;
    L_plasma = 2 * pi * R_plasma;
    V_plasma = 2 * pi^2 * R_plasma * a_plasma * b_plasma;
end

% PFC
coil_component = struct();
if enable_pfc
     pfc_list = {
         struct('name', 'PFC1', 'width', 5e-3, 'radius', 2.5e-3, ...
                 'distance', 0.200, 'z_values',    0.100, 'num_coils', 1), ...
```

```
        struct('name', 'PFC2', 'width', 5e-3, 'radius', 2.5e-3, ...
                'distance', 0.200, 'z_values', -0.100, 'num_coils', 1), ...
        struct('name', 'PFC3', 'width', 5e-3, 'radius', 2.5e-3, ...
                'distance', 0.100, 'z_values',    0.150, 'num_coils', 1), ...
        struct('name', 'PFC4', 'width', 5e-3, 'radius', 2.5e-3, ...
                'distance', 0.100, 'z_values', -0.150, 'num_coils', 1)
    };

    coil_component.PFC = struct();
    pfc_counter = 1;
    for k = 1:length(pfc_list)
        pfc = pfc_list{k};
        [pfc_x, ~, pfc_z] = setup_coil(pfc.num_coils, pfc.radius, pfc.width, ...
            pfc.distance, pfc.z_values, pfc.z_values, 0, 0);
        for i = 1:pfc.num_coils
            coil_component.PFC(pfc_counter).r = pfc_x(i);
            coil_component.PFC(pfc_counter).z = pfc_z(i);
            coil_component.PFC(pfc_counter).thickness = pfc.width;
            coil_component.PFC(pfc_counter).radius = pfc.radius;
            coil_component.PFC(pfc_counter).coil_length = 2 * pi * pfc.distance;
            pfc_counter = pfc_counter + 1;
        end
    end
end

%% ========== Plotting ==========

figure;
hold on;

if enable_chamber
    plot(outer_x, outer_z, 'ro', 'MarkerFaceColor', 'r', 'DisplayName', 'Outer Wall');
    plot(inner_x, inner_z, 'bo', 'MarkerFaceColor', 'b', 'DisplayName', 'Inner Wall');
end

if enable_plasma
    theta = linspace(0, 2*pi, 200);
```

```
    x_plasma = R_plasma + a_plasma * cos(theta);
    z_plasma = b_plasma * sin(theta);
    plot(x_plasma, z_plasma, 'k-', 'DisplayName', 'Plasma');
end

if enable_solenoid
    solenoid_z = linspace(-solenoid_height/2 + solenoid_total_radius, ...
                                    solenoid_height/2 - solenoid_total_radius,
solenoid_inner_num_coils)';
    plot(repmat(solenoid_inner_distance, solenoid_inner_num_coils, 1), solenoid_z, ...
        'gx', 'DisplayName', 'Solenoid Inner');
    if solenoid_layer_count == 2
        plot(repmat(solenoid_outer_distance, solenoid_outer_num_coils, 1),
solenoid_z, ...
            'mx', 'DisplayName', 'Solenoid Outer');
    end
end

if enable_pfc
    for p = 1:length(coil_component.PFC)
        plot(coil_component.PFC(p).r, coil_component.PFC(p).z, ...
            'go', 'MarkerSize', 6, 'MarkerFaceColor', 'g', 'DisplayName', 'PFC');
    end
end

xlabel('X Position (m)');
ylabel('Z Position (m)');
title('Chamber XZ Plane View');
legend('Location', 'bestoutside');
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 1.5, 'FontWeight', 'bold');
grid on;
hold off;
saveas(gcf, 'wall_rect_plot.png');

%% ========== Save to Excel ==========

% Chamber
```

```
if enable_chamber
    num_inner = length(inner_x);
    num_outer = length(outer_x);
    names = [arrayfun(@(i) sprintf('Inner Wall %d', i), 1:num_inner, 'UniformOutput',
false)'; ...
                arrayfun(@(i) sprintf('Outer Wall %d', i), 1:num_outer, 'UniformOutput',
false)'];
    r_values = [inner_x; outer_x];
    z_values = [inner_z; outer_z];
    thickness_values = [repmat(inner_params.thickness, num_inner, 1); outer_thk];
    radius_values = [repmat(inner_params.thickness/2, num_inner, 1); outer_r];
    coil_lengths = 2 * pi * r_values;

    chamber_table = table(names, r_values, z_values, thickness_values, radius_values,
coil_lengths, ...
        'VariableNames', {'Component', 'Distance_m', 'Z_Position_m', 'Thickness_m',
'Radius_m', 'Coil_Length_m'});
    save_table_with_overwrite(chamber_table, 'chamber_components.xlsx');
end

% Solenoid
if enable_solenoid
    solenoid_table = table({"solenoid"}, solenoid_radius, solenoid_radius_cooling,
isolation_thickness, solenoid_total_radius, ...
        solenoid_inner_num_coils, solenoid_outer_num_coils, solenoid_inner_distance,
solenoid_outer_distance, ...
        solenoid_height_between_turns, solenoid_height, solenoid_coil_A,
solenoid_a, ...
        'VariableNames', {'Component', 'Radius_m', 'Cooling_m', 'Isolation_Thick_m',
'Total_Radius_m', 'Inner_Num_Coils', 'Outer_Num_Coils', 'Inner_Distance_m',
'Outer_Distance_m', 'Height_Between_Turns_m', 'Total_Height_m', 'Coil_Area_m2',
'A_Param'});
    writetable(solenoid_table, 'solenoid_parameters.xlsx');
end

% Plasma
if enable_plasma
```

```
    plasma_table = table({"plasma"}, T_plasma, n_total_plasma, a_plasma, kappa,
R_plasma, A_plasma, L_plasma, V_plasma, ...
        'VariableNames', {'Component', 'Temperature_eV', 'Particle_Density_m3',
'Semi_Minor_Axis_m', 'Kappa', 'Major_Radius_m', 'Cross_Sectional_Area_m2',
'Length_m', 'Volume_m3'});
    writetable(plasma_table, 'plasma_parameters.xlsx');
end

% PFC
if enable_pfc
    coil_names = fieldnames(coil_component);
    coil_table_data = [];
    for i = 1:length(coil_names)
        comp_name = coil_names{i};
        coils = coil_component.(comp_name);
        for j = 1:length(coils)
            coil_table_data = [coil_table_data; {
                sprintf('%s %d', comp_name, j), ...
                coils(j).r, coils(j).z, coils(j).thickness, coils(j).radius,
coils(j).coil_length
            }];
        end
    end
    coil_table = cell2table(coil_table_data, ...
        'VariableNames', {'Component', 'Distance_m', 'Z_Position_m', 'Thickness_m',
'Radius_m', 'Coil_Length_m'});
    writetable(coil_table, 'coil_components.xlsx');
end

%% ====== Function Block ======

function [x_values, z_values, wall_struct] = generate_inner_wall(params)
    radius = params.thickness / 2;
    num_coils = ceil(params.height / params.thickness);
    z_values = linspace(-params.height/2 + radius, params.height/2 - radius, num_coils)';
    x_values = repmat(params.distance, num_coils, 1);
```

```
    wall_struct = repmat(struct('r',0,'z',0,'thickness',0,'radius',0,'coil_length',0), num_coils,
1);

    for i = 1:num_coils
        wall_struct(i).r = x_values(i);
        wall_struct(i).z = z_values(i);
        wall_struct(i).thickness = params.thickness;
        wall_struct(i).radius = radius;
        wall_struct(i).coil_length = 2 * pi * x_values(i);
    end
end

function [x_values, y_values, z_values] = setup_coil(num_coils, coil_radius,
coil_thickness, ...
    coil_distance, z_start, z_end, theta_start, theta_end, offset_x, use_sin_z)
    if nargin < 9, offset_x = 0; end
    if nargin < 10, use_sin_z = false; end
    theta_values = linspace(theta_start, theta_end, num_coils)';
    x_values = coil_distance * cos(theta_values) + offset_x;
    y_values = coil_distance * sin(theta_values);
    if use_sin_z
        z_values = coil_distance * sin(theta_values);
    else
        z_values = linspace(z_start, z_end, num_coils)';
    end
end

function save_table_with_overwrite(table_data, file_name)
    writetable(table_data, file_name);
    fprintf('File "%s" has been saved successfully (overwritten if existed).\n', file_name);
end
```

A.5 Value of Q_0 for mutual inductance between solenoid and ring coil (Table 27 in page 115 of Ref [10])

ρ^2	$\alpha = 0$	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	ρ^2
0	1	1	1	1	1	1	1	1	1	1	1	0
0.05	1	1.0000	1.0000	1.0000	1.0001	1.0001	1.0001	1.0001	1.0002	1.0002	1.0002	0.05
.10	1	1.0000	1.0000	1.0001	1.0002	1.0002	1.0003	1.0004	1.0006	1.0008	1.0009	.10
.15	1	1.0000	1.0001	1.0002	1.0004	1.0006	1.0008	1.0010	1.0013	1.0016	1.0020	.15
.20	1	1.0000	1.0002	1.0004	1.0006	1.0010	1.0014	1.0018	1.0024	1.0030	1.0036	.20
0.25	1	1.0001	1.0003	1.0005	1.0009	1.0014	1.0021	1.0029	1.0038	1.0046	1.0057	0.25
.30	1	1.0001	1.0004	1.0007	1.0013	1.0021	1.0030	1.0041	1.0053	1.0067	1.0082	.30
.35	1	1.0001	1.0005	1.0010	1.0018	1.0028	1.0041	1.0056	1.0072	1.0091	1.0111	.35
.40	1	1.0002	1.0006	1.0014	1.0024	1.0037	1.0054	1.0072	1.0094	1.0119	1.0146	.40
.45	1	1.0002	1.0008	1.0017	1.0030	1.0047	1.0068	1.0092	1.0120	1.0152	1.0186	.45
0.50	1	1.0002	1.0010	1.0021	1.0037	1.0058	1.0084	1.0114	1.0148	1.0188	1.0231	0.50
.55	1	1.0003	1.0012	1.0026	1.0045	1.0071	1.0102	1.0138	1.0180	1.0228	1.0282	.55
.60	1	1.0003	1.0014	1.0031	1.0054	1.0084	1.0122	1.0166	1.0216	1.0274	1.0338	.60
.65	1	1.0004	1.0016	1.0036	1.0064	1.0100	1.0144	1.0196	1.0256	1.0325	1.0402	.65
.70	1	1.0004	1.0018	1.0042	1.0074	1.0116	1.0167	1.0228	1.0300	1.0382	1.0473	.70
0.75	1	1.0005	1.0021	1.0048	1.0085	1.0133	1.0193	1.0264	1.0348	1.0444	1.0552	0.75
.80	1	1.0006	1.0024	1.0054	1.0097	1.0152	1.0221	1.0303	1.0400	1.0512	1.0640	.80
.85	1	1.0006	1.0027	1.0061	1.0110	1.0173	1.0251	1.0345	1.0458	1.0588	1.0739	.85
.90	1	1.0007	1.0030	1.0069	1.0124	1.0195	1.0284	1.0392	1.0522	1.0573	1.0850	.90
0.95	1	1.0008	1.0034	1.0077	1.0139	1.0218	1.0320	1.0442	1.0592	1.0768	1.0976	0.95
1.00	1	1.0009	1.0038	1.0086	1.0154	1.0244	1.0358	1.0498	1.0668	1.0873	1.1117	1.00

ρ^2	$\alpha = 0.50$	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	$\alpha = 1.00$	ρ^2
0	1	1	1	1	1	1	1	1	1	1	1	0
0.05	1.0002	1.0002	1.0003	1.0004	1.0005	1.0005	1.0006	1.0007	1.0008	1.0008	1.0009	0.05
.10	1.0009	1.0011	1.0013	1.0016	1.0018	1.0020	1.0023	1.0026	1.0029	1.0032	1.0035	.10
.15	1.0020	1.0024	1.0029	1.0034	1.0040	1.0046	1.0051	1.0057	1.0063	1.0070	1.0077	.15
.20	1.0036	1.0044	1.0052	1.0061	1.0070	1.0080	1.0090	1.0101	1.0112	1.0123	1.0135	.20
0.25	1.0057	1.0058	1.0081	1.0094	1.0108	1.0124	1.0140	1.0156	1.0172	1.0190	1.0208	0.25
.30	1.0082	1.0098	1.0116	1.0186	1.0156	1.0178	1.0200	1.0224	1.0248	1.0273	1.0299	.30
.35	1.0111	1.0134	1.0158	1.0185	1.0213	1.0242	1.0272	1.0304	1.0337	1.0372	1.0406	.35
.40	1.0146	1.0176	1.0208	1.0242	1.0279	1.0317	1.0358	1.0400	1.0443	1.0488	1.0534	.40
.45	1.0186	1.0224	1.0264	1.0308	1.0355	1.0404	1.0456	1.0510	1.0566	1.0623	1.0683	.45
0.50	1.0231	1.0278	1.0330	1.0384	1.0444	1.0506	1.0571	1.0640	1.0710	1.0784	1.0858	0.50
.55	1.0282	1.0340	1.0404	1.0471	1.0544	1.0622	1.0704	1.0790	1.0878	1.0970	1.1064	.55
.60	1.0338	1.0410	1.0487	1.0570	1.0860	1.0756	1.0857	1.0964	1.1076	1.1189	1.1305	.60
.65	1.0402	1.0488	1.0582	1.0583	1.0792	1.0910	1.1036	1.1168	1.1304	1.1446	1.1593	.65
.70	1.0473	1.0575	1.0688	1.0812	1.0945	1.1089	1.1242	1.1405	1.1576	1.1754	1.1941	.70
0.75	1.0552	1.0674	1.0810	1.0960	1.1124	1.1301	1.1491	1.1693	1.1908	1.2134	1.2372	0.75
.80	1.0640	1.0785	1.0947	1.1126	1.1324	1.1545	1.1786	1.2044	1.2320	1.2612	1.2922	.80
.85	1.0739	1.0911	1.1106	1.1324	1.1568	1.1843	1.2150	1.2490	1.2864			.85
.90	1.0850	1.1054	1.1289	1.1558	1.1866	1.2220	1.2623	1.3080	1.3589	1.4150	1.4763	.90
0.95	1.0976	1.1216	1.1502	1.1842	1.2245							0.95
1.00	1.1117	1.1403	1.1752	1.2157	1.2733	1.3630	1.4978		1.7430		∞	1.00

A.6 Value of f for mutual inductance between ring coils. (Table 13 in page 79 of Ref [10])

k'^2	f	Diff.	$\log f$	Diff.	k'^2	f	Diff.	$\log f$	Diff.
0.010	0.021474		$\bar{2}.33191$		0.260	0.003805		3.58034	
		−4159		−9349			− 156		−1819
.020	.017315		.23842		.270	.003649		.56215	
		−2378		−6596			− 149		−1805
.030	.014937		.17246		.280	.003500		.54410	
		−1653		−4913			− 141		−1792
.040	.013284		.12333		.290	.003359		.52618	
		−1258		−4319			− 135		−1783
0.050	0.012026		$\bar{2}.08014$		0.300	0.003224		3.50835	
		−1009		−3807			− 129		−1773
.060	.011017		.04207		.310	.003095		.49062	
		− 838		−3437			− 124		−1767
.070	.010179		$\bar{2}.00770$		.320	.002971		.47295	
		− 715		−3162			− 118		−1760
.080	.009464		$\bar{3}.97608$		.330	.002853		.45535	
		− 621		−2946			− 113		−1757
.090	.008843		.94662		.340	.002740		.43778	
		− 546		−2772			− 108		−1754
0.100	0.008297		$\bar{3}.91890$		0.350	0.0026317		$\bar{3}.42024$	
		− 487		−2627			−1041		−1753
.110	.007810		.89263		.360	.0025276		.40271	
		− 439		−2509			−1000		−1753
.120	.007371		.86754		.370	.0024276		.38518	
		− 397		−2407			− 961		−1754
.130	.006974		.84347		.380	.0023315		.36764	
		− 363		−2321			− 924		−1756
.140	.006611		.82026		.390	.0022391		.35008	
		− 333		−2246			− 889		−1760
0.150	0.006278		$\bar{3}.79780$		0.400	0.0021502		$\bar{3}.33248$	
		− 308		−2181			− 856		−1765
.160	.005970		.77599		.410	.0020646		.31483	
		− 285		−2124			− 825		−1769
.170	.005685		.75475		.420	.0019821		.29712	
		− 265		−2074			− 795		−1778
.180	.005420		.73401		.430	.0019026		.27934	
		− 247		−2030			− 767		−1786
.190	.005173		.71371		.440	.0018259		.26148	
		− 232		−1991			− 740		−1796
0.200	0.004941		$\bar{3}.69380$		0.450	0.0017519		$\bar{3}.24352$	
		− 218		−1957			− 714		−1807
.210	.004723		.67423		.460	.0016805		.22545	
		− 205		−1926			− 689		−1819
.220	.004518		.65497		.470	.0016116		.20726	
		− 193		−1899			− 665		−1832
.230	.004325		.63598		.480	.0015451		.18894	
		− 183		−1875			− 643		−1846
.240	.004142		.61723		.490	.0014808		.17048	
		− 173		−1854			− 622		−1862
0.250	0.003969		$\bar{3}.59869$		0.500	0.0014186		$\bar{3}.15186$	
		− 164		−1835			− 601		−1879

k'^2	f	Diff.	$\log f$	Diff.	k'^2	f	Diff.	$\log f$	Diff.
0.500	0.0014186		$\bar{3}.15186$		0.750	0.0003805		$\bar{4}.58033$	
		−601		−1879			− 260		−3068
.510	.0013585		.13307		.760	.0003545		.54965	
		−581		−1898			− 250		−3177
.520	.0013004		.11409		.770	.0003295		.51788	
		−561		−1917			− 241		−3296
.530	.0012443		.09492		.780	.0003054		.48492	
		−543		−1939			− 231		−3427
.540	.0011900		.07553		.790	.0002823		.45065	
		−526		−1962			− 223		−3570
0.550	0.0011374		$\bar{3}.05591$		0.800	0.00025998		$\bar{4}.41495$	
		−509		−1987			−2139		−3730
.560	.0010865		.03604		.810	.00023859		.37765	
		−492		−2012			−2053		−3906
.570	.0010373		$\bar{3}.01592$		.820	.00021806		.33859	
		−476		−2041			−1966		−4105
.580	0.0009897		$\bar{4}.99551$		.830	.00019840		.29754	
		−461		−2071			−1881		−4326
.590	.0009436		.97480		.840	.00017959		.25428	
		−446		−2103			−1797		−4577
0.600	0.0008990		$\bar{4}.95377$		0.850	0.00016162		$\bar{4}.20851$	
		−432		−2137			−1712		−4867
.610	.0008558		.93240		.860	.00014450		.15986	
		−417		−2174			−1629		−5194
.620	.0008141		.91066		.870	.00012821		.10792	
		−405		−2213			−1545		−5577
.630	.0007736		.88853		.880	0.00011276		$\bar{4}.05215$	
		−391		−2254			−1461		−6028
.640	.0007345		.86599		.890	.00009815		$\bar{5}.99187$	
		−379		−2299			−1377		−6565
0.650	0.0006966		$\bar{4}.84300$		0.900	0.00008438		$\bar{5}.92622$	
		−366		−2346			−1292		
.660	.0006600		.81954		.910	.00007146		.85405	
		−354		−2398			−1206		
.670	.0006246		.79556		.920	.00005940		.77382	
		−343		−2451			−1116		
.680	.0005903		.77105		.930	.00004824		.68336	
		−332		−2510			−1026		
.690	.0005571		.74595		.940	.00003798		.57950	
		−320		−2573			− 932		
0.700	0.0005251		$\bar{4}.72022$		0.950	0.00002866		$\bar{5}.45732$	
		−310		−2640			− 831		
.710	.0004941		.69382		.960	.00002035		.30858	
		−299		−2714			− 723		
.720	.0004642		.66668		.970	0.00001312		$\bar{5}.11782$	
		−289		−2791			− 604		
.730	.0004353		.63877		.980	.00000708		$\bar{6}.85035$	
		−279		−2876			− 459		
.740	.0004074		.61001		0.990	0.00000249		$\bar{6}.39551$	
		−269		−2968			− 249		
0.750	0.0003805		$\bar{4}.58033$		1.000	0			
		−260		−3068					

A.7 Value of f for mutual inductance between ring coils. (Table 14 in page 81 of Ref [10])

$\log k'^2$	f	Diff.	$\log k'^2$	f	Diff.	$\log k'^2$	f	Diff.
$\bar{6}.0$	0.079093		$\bar{4}.0$	0.050163		$\bar{2}.0$	0.021478	
		−1446			−1446			−1394
$\bar{6}.1$	.077647		$\bar{4}.1$	.048717		$\bar{2}.1$	.020084	
		−1447			−1445			−1384
$\bar{6}.2$	.076200		$\bar{4}.2$	.047272		$\bar{2}.2$	.018700	
		−1447			−1445			−1371
$\bar{6}.3$	.074753		$\bar{4}.3$	.045827		$\bar{2}.3$	.017329	
		−1447			−1445			−1357
$\bar{6}.4$	.073306		$\bar{4}.4$	.044382		$\bar{2}.4$	.015972	
		−1446			−1444			−1340
$\bar{6}.5$	0.071860		$\bar{4}.5$	0.042938		$\bar{2}.5$	0.014632	
		−1447			−1444			−1321
$\bar{6}.6$	.070413		$\bar{4}.6$	.041494		$\bar{2}.6$	.013311	
		−1447			−1443			−1298
$\bar{6}.7$	.068966		$\bar{4}.7$	.040051		$\bar{2}.7$	.012013	
		−1446			−1443			−1271
$\bar{6}.8$	.067520		$\bar{4}.8$	.038608		$\bar{2}.8$	0.010742	
		−1447			−1441			−1240
$\bar{6}.9$	.066073		$\bar{4}.9$	.037167		$\bar{2}.9$	.009502	
		−1446			−1440			−1205
$\bar{5}.0$	0.064626		$\bar{3}.0$	0.035727		$\bar{1}.0$	0.008297	
		−1447			−1439			
$\bar{5}.1$	.063180		$\bar{3}.1$	.034288				
		−1447			−1437			
$\bar{5}.2$	.061733		$\bar{3}.2$	.032851				
		−1446			−1435			
$\bar{5}.3$	.060287		$\bar{3}.3$	.031416				
		−1447			−1432			
$\bar{5}.4$	.058840		$\bar{3}.4$	.029984				
		−1446			−1430			
$\bar{5}.5$	0.057394		$\bar{3}.5$	0.028554				
		−1447			−1426			
$\bar{5}.6$	.055947		$\bar{3}.6$	.027128				
		−1447			−1421			
$\bar{5}.7$	.054500		$\bar{3}.7$	.025707				
		−1445			−1416			
$\bar{5}.8$	.053055		$\bar{3}.8$	.024291				
		−1446			−1410			
$\bar{5}.9$	.051609		$\bar{3}.9$	.022881				
		−1446			−1403			
$\bar{4}.0$	0.050163		$\bar{2}.0$	0.021478				

A.8 Value of f for mutual inductance between ring coils. (Table 15 in page 82 of Ref [10])

$\log k^2$	$\log f$	Diff. d_1	Diff. d_2	$\log k^2$	$\log f$	Diff. d_1	Diff. d_2
$\bar{4}.0$	$\bar{9}.39227$			$\bar{3}.5$	$\bar{7}.64327$		
		15001				15027	
$\bar{4}.1$	.54228			$\bar{3}.6$	.79354		7
		15001				15034	
$\bar{4}.2$	.69229			$\bar{3}.7$	$\bar{7}.94388$		8
		15001				15042	
$\bar{4}.3$	.85230			$\bar{3}.8$	$\bar{6}.09430$		12
		15002				15054	
$\bar{4}.4$	$\bar{9}.99232$			$\bar{3}.9$	.24484		13
		15002				15067	
$\bar{4}.5$	$\bar{8}.14234$			$\bar{2}.0$	.39551		18
		15003				15085	
$\bar{4}.6$	.29237			$\bar{2}.1$	.54636		23
		15003				15108	
$\bar{4}.7$	.44240		1	$\bar{2}.2$	.69744		27
		15004				15135	
$\bar{4}.8$	.59244		2	$\bar{2}.3$	$\bar{6}.84879$		37
		15006				15172	
$\bar{4}.9$	.74250		1	$\bar{2}.4$	$\bar{5}.00051$		45
		15007				15217	
$\bar{3}.0$	$\bar{8}.89257$		1	$\bar{2}.5$	.15268		57
		15008				15274	
$\bar{3}.1$	$\bar{7}.04265$		3	$\bar{2}.6$	.30542		75
		15011				15349	
$\bar{3}.2$	.19276		2	$\bar{2}.7$	.45891		94
		15013				15443	
$\bar{3}.3$	.34289		4	$\bar{2}.8$	.61334		118
		15017				15565	
$\bar{3}.4$	.49306		4	$\bar{2}.9$	.76899		158
		15021				15723	
$\bar{3}.5$	$\bar{7}.64327$		6	$\bar{1}.0$	$\bar{5}.92622$		
		15027					

A.9 Value of f for mutual inductance between ring coils. (Table 16 in page 83 of Ref [10])

δ	f	Diff.	δ	f	Diff.	δ	f	Diff.	δ	f	Diff.
0.01	0.05016		0.26	0.010723		0.51	0.004800		0.76	0.0024659	
		− 869			− 383			− 136			− 599
.02	.04147		.27	.010340		.52	.004664		.77	.0024060	
		− 508			− 366			− 132			− 581
.03	.03639		.28	.009974		.53	.004532		.78	.0023479	
		− 359			− 347			− 127			− 563
.04	.03280		.29	.009627		.54	.004405		.79	.0022916	
		− 277			− 331			− 122			− 547
0.05	0.03003		0.30	0.009296		0.55	0.004283		0.80	0.0022369	
		− 226			− 314			− 118			− 531
.05	.02777		.31	.008980		.56	.004165		.81	.0021838	
		− 189			− 301			− 114			− 515
.07	.02588		.32	.008679		.57	.004051		.82	.0021323	
		− 164			− 289			− 111			− 500
.08	.02424		.33	.008390		.58	.003940		.83	.0020823	
		− 143			− 276			− 106			− 486
.09	.02281		.34	.008114		.59	.003834		.84	.0020337	
		− 127			− 264			− 103			− 472
0.10	0.021539		0.35	0.007850		0.60	0.003730		0.85	0.0019865	
		− 1143			− 253			− 99			− 458
.11	.020396		.36	.007597		.61	.003631		.86	.0019407	
		− 1035			− 243			− 97			− 445
.12	.019361		.37	.007354		.62	.003534		.87	.0018962	
		− 944			− 233			− 93			− 432
.13	.018417		.38	.007121		.63	.003441		.88	.0018530	
		− 867			− 223			− 90			− 421
.14	.017550		.39	.006898		.64	.003351		.89	.0018109	
		− 800			− 214			− 88			− 408
0.15	0.016750		0.40	0.006684		0.65	0.003263		0.90	0.0017701	
		− 741			− 207			− 84			− 397
.16	.016009		.41	.006477		.66	.003179		.91	.0017304	
		− 690			− 198			− 82			− 386
.17	.015319		.42	.006279		.67	.003097		.92	.0016918	
		− 643			− 190			− 79			− 376
.18	.014676		.43	.006089		.68	.003018		.93	.0016542	
		− 603			− 183			− 77			− 364
.19	.014073		.44	.005906		.69	.002941		.94	.0016178	
		− 566			− 176			− 75			− 356
0.20	0.013507		0.45	0.005730		0.70	0.002866		0.95	0.0015822	
		− 532			− 170			− 72			− 345
.21	.012975		.46	.005560		.71	.002794		.96	.0015477	
		− 502			− 164			− 70			− 336
.22	.012473		.47	.005396		.72	.002725		.97	.0015141	
		− 473			− 157			− 68			− 327
.23	.012000		.48	.005239		.73	.002657		.98	.0014814	
		− 449			− 152			− 66			− 318
.24	.011551		.49	.005087		.74	.002591		0.99	.0014496	
		− 425			− 146			− 63			− 310
0.25	0.011126		0.50	0.004941		0.75	0.002528		1.00	0.0014186	
		− 403			− 141			− 62			

A.10 Value of f for mutual inductance between ring coils. (Table 17 in page 84 of Ref [10])

Δ	f	Diff.	Δ	f	Diff.	Δ	f	Diff.	Δ	f	Diff.
1.00	0.0014186		0.75	0.0007345		0.50	0.00025999		0.25	0.00003683	
		−304			−235			−1377			− 413
0.99	.0013882		.74	.0007110		.49	.00024622		.24	.00003270	
		−302			−231			−1335			− 382
.98	.0013579		.73	.0006879		.48	.00023287		.23	.00002888	
		−300			−228			−1293			− 353
.97	.0013279		.72	.0006651		.47	.00021994		.22	.00002535	
		−297			−224			−1251			− 323
.96	.0012982		.71	.0006427		.46	.00020743		.21	.00002212	
		−296			−221			−1210			− 296
0.95	0.0012686		0.70	0.0006206		0.45	0.00019533		0.20	0.000019165	
		−293			−217			−1168			−2687
.94	.0012393		.69	.0005989		.44	.00018365		.19	.000016478	
		−290			−214			−1126			−2429
.93	.0012103		.68	.0005775		.43	.00017239		.18	.000014049	
		−288			−210			−1085			−2184
.92	.0011814		.67	.0005565		.42	.00016154		.17	.000011865	
		−286			−206			−1044			−1949
.91	.0011529		.66	.0005359		.41	.00015109		.16	.000009916	
		−283			−202			−1003			−1827
0.90	0.0011246		0.65	0.0005157		0.40	0.00014106		0.15	0.000008189	
		−280			−198			− 963			−1517
.89	.0010966		.64	.0004959		.39	.00013143		.14	.000006672	
		−278			−195			− 922			−1319
.88	.0010688		.63	.0004764		.38	.00012221		.13	.000005353	
		−275			−191			− 883			−1135
.87	.0010413		.62	.0004573		.37	.00011338		.12	.000004218	
		−272			−186			− 843			− 963
.86	.0010141		.61	.0004387		.36	0.00010495		.11	.000003255	
		−270			−183			− 803			− 806
0.85	0.0009871		0.60	0.0004204		0.35	0.00009692		0.10	0.000002449	
		−266			−179			− 766			− 661
.84	.0009605		.59	.0004025		.34	.00008926		.09	.0000017S8	
		−264			−175			− 726			− 531
.83	.0009341		.58	.0003850		.33	.00008200		.08	.000001257	
		−260			−170			− 710			− 414
.82	.0009081		.57	.0003680		.32	.00007510		.07	.000000843	
		−258			−167			− 652			− 311
.81	.0008823		.56	.0003513		.31	.00006858		.06	.000000532	
		−254			−163			− 616			− 224
0.80	0.0008569		0.55	0.0003350		0.30	0.00006242		0.05	.000000308	
		−251			−158			− 580			
.79	.0008318		.54	.0003192		.29	.00005662				
		−248			−154			− 546			
.78	.0008070		.53	.0003038		.28	.00005116				
		−245			−150			− 511			
.77	.0007825		.52	.0002888		.27	.00004605				
		−242			−146			− 477			
.76	.0007583		.51	.0002742		.26	.00004128				
		−238			−142			− 445			
0.75	0.0007345		0.50	0.0002600		0.25	0.00003683				

A.11 Resistance and inductance calculation code

```
% The code is in Student_Thesis/2025_ypi/Code/R_and_M_thesis on the NAS.
enable_plasma = false;

% === Read components data ===

% Read chamber components data
chamber_data = readtable('chamber_components.xlsx');

% Extract the relevant data from the table
component_names = chamber_data.Component;                % Component names
r_values = chamber_data.Distance_m;                      % Distance (r) values
z_values = chamber_data.Z_Position_m;                    % Z-position values
thickness_values = chamber_data.Thickness_m;             % Thickness values
radius_values = chamber_data.Radius_m;                   % Radius values
coil_length_values = chamber_data.Coil_Length_m;         % Coil length values

% Initialize counters for inner and outer walls
num_inner_coils = 0;
num_outer_coils = 0;

% Initialize chamber components structures
chamber_components.inner_wall = struct();
chamber_components.outer_wall = struct();

% Iterate through each component and classify into inner or outer wall
for i = 1:height(chamber_data)
    component_name = component_names{i};

    if contains(component_name, 'Inner Wall')
        % Increment the inner wall coil counter
        num_inner_coils = num_inner_coils + 1;

        % Assign values to the inner wall structure
        chamber_components.inner_wall(num_inner_coils).r = r_values(i);
        chamber_components.inner_wall(num_inner_coils).z = z_values(i);
```

```
            chamber_components.inner_wall(num_inner_coils).thickness =
thickness_values(i);
            chamber_components.inner_wall(num_inner_coils).radius = radius_values(i);
            chamber_components.inner_wall(num_inner_coils).coil_length =
coil_length_values(i);

        elseif contains(component_name, 'Outer Wall')
            % Increment the outer wall coil counter
            num_outer_coils = num_outer_coils + 1;

            % Assign values to the outer wall structure
            chamber_components.outer_wall(num_outer_coils).r = r_values(i);
            chamber_components.outer_wall(num_outer_coils).z = z_values(i);
            chamber_components.outer_wall(num_outer_coils).thickness =
thickness_values(i);
            chamber_components.outer_wall(num_outer_coils).radius = radius_values(i);
            chamber_components.outer_wall(num_outer_coils).coil_length =
coil_length_values(i);
        end
end

% Display the results
fprintf('Number of inner wall coils: %d\n', num_inner_coils);
fprintf('Number of outer wall coils: %d\n', num_outer_coils);

% Read solenoid data
solenoid_data = readtable('solenoid_parameters.xlsx');

% Extract the relevant parameters for the solenoid
solenoid_radius = solenoid_data.Radius_m;                    % Radius of the solenoid
isolation_thick = solenoid_data.Isolation_Thick_m;           % Thickness of isolation
layer
solenoid_total_radius = solenoid_data.Total_Radius_m;        % Total radius of solenoid
solenoid_inner_num_coils = solenoid_data.Inner_Num_Coils;    % Number of inner coils
solenoid_outer_num_coils = solenoid_data.Outer_Num_Coils;    % Number of outer coils
solenoid_inner_distance = solenoid_data.Inner_Distance_m;    % Inner coil distance from
center
```

```
solenoid_outer_distance = solenoid_data.Outer_Distance_m;    % Outer coil distance from center
solenoid_height_between_turns = solenoid_data.Height_Between_Turns_m; % Height between turns
solenoid_height = solenoid_data.Total_Height_m;                % Total height of solenoid
solenoid_coil_A = solenoid_data.Coil_Area_m2;                  % Cross-sectional area of solenoid
solenoid_a = solenoid_data.A_Param;                            % Solenoid parameter 'a'

% Read plasma data
if enable_plasma
    plasma_data = readtable('plasma_parameters.xlsx');

    % Assign the values back to their original variable names
    T_plasma = plasma_data.Temperature_eV(1);                  % Initial temperature (K)
    n_total_plasma = plasma_data.Particle_Density_m3(1);       % Total particle density (m^-3)
    a_plasma = plasma_data.Semi_Minor_Axis_m(1);               % Plasma cross-section semi-minor axis (m)
    kappa = plasma_data.Kappa(1);                              % Plasma cross-section semi-major axis (m)
    b_plasma = a_plasma * kappa;
    R_plasma = plasma_data.Major_Radius_m(1);                  % Major radius (m)
    A_plasma = plasma_data.Cross_Sectional_Area_m2(1);         % Plasma cross-sectional area (m^2)
    L_plasma = plasma_data.Length_m(1);
    V_plasma = plasma_data.Volume_m3(1);                       % Plasma volume (m^3) for toroidal geometry
    li_plasma = 0.5;
    mu0 = 4 * pi * 1e-7;        % Permeability of vacuum (H/m)
    Lp = mu0 * R_plasma * (log(8 * R_plasma / a_plasma) + li_plasma / 2 - 2);
    Rp = 0;
```

```
    total_num_coils = num_inner_coils + num_outer_coils + 1 + 1; % +1 for solenoid, +1
for plasma

elseif ~enable_plasma
    total_num_coils = num_inner_coils + num_outer_coils + 1; % +1 for solenoid
end

% === Collect all components' data ===
coils_distance = zeros(total_num_coils, 1);
coils_z_values = zeros(total_num_coils, 1);
coils_thickness = zeros(total_num_coils, 1);
coils_num = zeros(total_num_coils, 1);

% Solenoid (index 1)
index = 1; % In case PFC need to put in front of CS
coils_distance(index) = solenoid_inner_distance; % For mutual inductance calculations,
we can use inner distance
coils_z_values(index) = 0; % Assuming solenoid is centered at z=0
coils_thickness(index) = solenoid_radius * 2; % Diameter
coils_num(index) = 1;

% Inner wall coils (indices 2 to num_inner_coils+1)
for i = 1:num_inner_coils
    coils_distance(i+index) = chamber_components.inner_wall(i).r;
    coils_z_values(i+index) = chamber_components.inner_wall(i).z;
    coils_thickness(i+index) = chamber_components.inner_wall(i).thickness;
    coils_num(i+index) = num_inner_coils;
end

% Outer wall coils (indices num_inner_coils+2 to 1+num_inner_coils+num_outer_coils)
for i = 1:num_outer_coils
    idx = i + num_inner_coils + index;
    coils_distance(idx) = chamber_components.outer_wall(i).r;
    coils_z_values(idx) = chamber_components.outer_wall(i).z;
    coils_thickness(idx) = chamber_components.outer_wall(i).thickness;
    coils_num(idx) = num_outer_coils;
end
```

```
% Material properties
rho_copper = 1.72e-8;                    % Resistivity of copper [Ohm·m]
chamber_resistivity = 6.9e-7;            % Resistivity of chamber material [Ohm·m] Stainless steel 304
copper_permeability = 4e-7*pi;           % Permeability of copper [H/m]
chamber_permeability = 1.25663706212e-6 * 1.008; % Permeability of chamber material [H/m]

current_idx = 1;
% Initialize resistance and inductance matrices
R = zeros(total_num_coils, total_num_coils);
M = zeros(total_num_coils, total_num_coils);

% Resistance calculations for solenoid
solenoid_inner_coil_length = 2 * pi * solenoid_inner_distance;
solenoid_outer_coil_length = 2 * pi * solenoid_outer_distance;
solenoid_inner_resistance = rho_copper * solenoid_inner_coil_length / solenoid_coil_A * solenoid_inner_num_coils;
solenoid_outer_resistance = rho_copper * solenoid_outer_coil_length / solenoid_coil_A * solenoid_outer_num_coils;
solenoid_resistance = solenoid_inner_resistance + solenoid_outer_resistance;

% Inductance calculation for solenoid
solenoid_inductance = copper_permeability * (solenoid_inner_num_coils + solenoid_outer_num_coils)^2 * solenoid_a / solenoid_height;

% Place solenoid resistance and inductance in the matrices
R(current_idx,current_idx) = solenoid_resistance;
M(current_idx,current_idx) = solenoid_inductance;

% Initialize current index after solenoid
current_idx = 2; % Solenoid is at index 1

% Calculate resistance and inductance for inner wall components
[R, M] = calculate_self_inductance_and_resistance(chamber_components.inner_wall, current_idx, chamber_resistivity, R, M);
```

```
% Update current index
current_idx = current_idx + num_inner_coils;

% Calculate resistance and inductance for outer wall components
[R, M] = calculate_self_inductance_and_resistance(chamber_components.outer_wall,
current_idx, chamber_resistivity, R, M);

if enable_plasma
    current_idx = current_idx + num_outer_coils;
    % Plasma
    coils_distance(current_idx) = R_plasma;
    coils_z_values(current_idx) = 0;
    coils_thickness(current_idx) = 0;
    coils_num(current_idx) = 1;

    R(current_idx, current_idx) = Rp;
    M(current_idx, current_idx) = Lp;
end

% Function to calculate self-inductance and resistance
function [R, M] = calculate_self_inductance_and_resistance(components, start_idx,
resistivity, R, M)
    % Function to calculate self-inductance and resistance for a set of coils
    %
    % Inputs:
    %     components    - Array of component structures with fields:
    %                         .r (radius), .thickness, .radius (coil radius), .coil_length
    %     start_idx     - Starting index in the R and M matrices
    %     resistivity - Resistivity of the material [Ohm·m]
    %     R                 - Resistance matrix to update
    %     M                 - Inductance matrix to update
    %
    % Outputs:
    %     R                 - Updated resistance matrix
    %     M                 - Updated inductance matrix
```

```
    num_coils = length(components);
    for i = 1:num_coils
        % Resistance calculation
        coil_length = components(i).coil_length; % Circumference [m]
        cross_sectional_area = components(i).radius^2 * pi; % [m^2]
        resistance = resistivity * coil_length / cross_sectional_area;
        idx = start_idx + i - 1;
        R(idx, idx) = resistance;

        % Inductance calculation
        a_constant = components(i).r * 100; % Convert to cm
        c_constant = components(i).thickness * 100; % Convert to cm
        c_over_2a = c_constant / (2 * a_constant);
        P_prim = 4 * pi * (((1 + (c_over_2a^2) / 6)*(log(8 / (c_over_2a^2)) /
log(exp(1))) * (1 / 2)) - 0.84834 + 0.2041 * (c_over_2a^2));
        inductance = 0.001 * (components(i).r - components(i).radius) * 100 * P_prim *
1e-6; % [H]
        M(idx, idx) = inductance;
    end
end

% === Mutual inductance calculations ===

% Load mutual inductance tables from Excel files
solenoid_mutual_table = xlsread("table27.xlsx");
table13 = xlsread("table13.xlsx");
table14 = xlsread("table14.xlsx");
table15 = xlsread("table15.xlsx");
table16 = xlsread("table16.xlsx");
table17 = xlsread("table17.xlsx");

% Preprocess data from tables
alpha_values = solenoid_mutual_table(1, 2:end);
rho_squared_values = solenoid_mutual_table(2:end, 1);
solenoid_q_values = solenoid_mutual_table(2:end, 2:end);

q_values = table16(:, 1);
```

```
f_values = table16(:, 2);
t_values = table17(:, 1);
k_values = table17(:, 2);

k_prim_square_values_table13 = table13(:, 1);
f_values_table13 = table13(:, 2);

log_k_prim_square_values_table14 = table14(:, 1);
f_values_table14 = table14(:, 2);

log_k_square_values_table15 = table15(:, 1);
log_f_values_table15 = table15(:, 2);

solenoid_top_z = solenoid_height / 2;
solenoid_bottom_z = -solenoid_height / 2;

% Mutual inductance between solenoid and ring components
for i = 2:total_num_coils
    % Alpha parameter
    alpha_solenoid = solenoid_inner_distance / coils_distance(i);

    % Compute D and d based on relative positions
    z_coil = coils_z_values(i);

    if z_coil > solenoid_top_z
        D = z_coil - solenoid_bottom_z;
        d = z_coil - solenoid_top_z;
        N_D = solenoid_inner_num_coils * 2;
        N_d = 1;

        % Calculate rho squared values
        rho_squared_D = coils_distance(i)^2 / (coils_distance(i)^2 + D^2);
        rho_squared_d = coils_distance(i)^2 / (coils_distance(i)^2 + d^2);

        % Interpolate q values from solenoid_mutual_table
        q_solenoid_D = interp2(alpha_values, rho_squared_values, solenoid_q_values,
alpha_solenoid, rho_squared_D, 'linear', 0);
```

```
        q_solenoid_d = interp2(alpha_values, rho_squared_values, solenoid_q_values, alpha_solenoid, rho_squared_d, 'linear', 0);

        % Mutual inductance calculation
        M_D = 0.002 * pi^2 * solenoid_outer_distance * 100 * alpha_solenoid * sqrt(rho_squared_D) * N_D * q_solenoid_D * 1e-6;
        M_d = 0.002 * pi^2 * solenoid_outer_distance * 100 * alpha_solenoid * sqrt(rho_squared_d) * N_d * q_solenoid_d * 1e-6;

        % Compute mutual inductance between solenoid and coil i
        M(1, i) = M_D - M_d;
        M(i, 1) = M(1, i); % Symmetric matrix

    elseif z_coil < solenoid_bottom_z
        D = solenoid_top_z - z_coil;
        d = solenoid_bottom_z - z_coil;
        N_D = solenoid_inner_num_coils * 2;
        N_d = 1;

        % Calculate rho squared values
        rho_squared_D = coils_distance(i)^2 / (coils_distance(i)^2 + D^2);
        rho_squared_d = coils_distance(i)^2 / (coils_distance(i)^2 + d^2);

        % Interpolate q values from solenoid_mutual_table
        q_solenoid_D = interp2(alpha_values, rho_squared_values, solenoid_q_values, alpha_solenoid, rho_squared_D, 'linear', 0);
        q_solenoid_d = interp2(alpha_values, rho_squared_values, solenoid_q_values, alpha_solenoid, rho_squared_d, 'linear', 0);

        % Mutual inductance calculation
        M_D = 0.002 * pi^2 * solenoid_outer_distance * 100 * alpha_solenoid * sqrt(rho_squared_D) * N_D * q_solenoid_D * 1e-6;
        M_d = 0.002 * pi^2 * solenoid_outer_distance * 100 * alpha_solenoid * sqrt(rho_squared_d) * N_d * q_solenoid_d * 1e-6;

        % Compute mutual inductance between solenoid and coil i
        M(1, i) = M_D - M_d;
```

```
            M(i, 1) = M(1, i); % Symmetric matrix

        elseif z_coil == solenoid_bottom_z || z_coil == solenoid_top_z

            D = solenoid_height;
            N_D = solenoid_inner_num_coils * 2;

            % Calculate rho squared values
            rho_squared_D = coils_distance(i)^2 / (coils_distance(i)^2 + D^2);

            % Interpolate q values from solenoid_mutual_table
            q_solenoid_D = interp2(alpha_values, rho_squared_values, solenoid_q_values,
alpha_solenoid, rho_squared_D, 'linear', 0);

            % Mutual inductance calculation
            M_D = 0.002 * pi^2 * solenoid_outer_distance * 100 * alpha_solenoid *
sqrt(rho_squared_D) * N_D * q_solenoid_D * 1e-6;

            % Compute mutual inductance between solenoid and coil i
            M(1, i) = M_D;
            M(i, 1) = M(1, i); % Symmetric matrix

        else
            % Coil is within the solenoid range
            solenoid_z_difference_top = solenoid_top_z - z_coil;
            solenoid_z_difference_bottom = z_coil - solenoid_bottom_z;
            N_top = (solenoid_z_difference_top / (solenoid_total_radius * 2)) * 2;
            N_bottom = solenoid_inner_num_coils*2 - N_top;
            N_D = N_top;
            N_d = N_bottom;
            D = solenoid_z_difference_top;
            d = solenoid_z_difference_bottom;

            % Calculate rho squared values
            rho_squared_D = coils_distance(i)^2 / (coils_distance(i)^2 + D^2);
            rho_squared_d = coils_distance(i)^2 / (coils_distance(i)^2 + d^2);
```

```
            % Interpolate q values from solenoid_mutual_table
            q_solenoid_D = interp2(alpha_values, rho_squared_values, solenoid_q_values,
alpha_solenoid, rho_squared_D, 'linear', 0);
            q_solenoid_d = interp2(alpha_values, rho_squared_values, solenoid_q_values,
alpha_solenoid, rho_squared_d, 'linear', 0);

            % Mutual inductance calculation
            M_D = 0.002 * pi^2 * solenoid_outer_distance * 100 * alpha_solenoid *
sqrt(rho_squared_D) * N_D * q_solenoid_D * 1e-6;
            M_d = 0.002 * pi^2 * solenoid_outer_distance * 100 * alpha_solenoid *
sqrt(rho_squared_d) * N_d * q_solenoid_d * 1e-6;

            % Compute mutual inductance between solenoid and coil i
            M(1, i) = M_D + M_d;
            M(i, 1) = M(1, i); % Symmetric matrix
        end
end

% Mutual inductance between each ring components
for i = 2:total_num_coils
        for j = (i+1):total_num_coils
            if i ~= j
                d = abs(coils_z_values(i) - coils_z_values(j)) * 100; % Convert to cm

                if coils_distance(i) == coils_distance(j) && coils_thickness(i) ==
coils_thickness(j)
                    % Coaxial coils of equal size
                    %r = coils_distance(i) * 100; % Radius in cm
                    r = coils_distance(i) * 10^2 * (1 + (coils_thickness(i) * 10^2)^2 / (24 *
(coils_thickness(i) * 10^2)^2));
                    radio = d / (2 * r);

                    if radio <= 1
                        f = interp1(q_values, f_values, radio, 'linear', 'extrap');
                        M_ij = coils_distance(i) * 100 * f * 1e-6; % Mutual inductance in
H
                    else
```

```
                radio_inv = 1 / radio;
                f = interp1(t_values, k_values, radio_inv, 'linear', 'extrap');
                M_ij = coils_distance(i) * 100 * f * 1e-6; % Mutual inductance in
H
            end
        else
            % Coaxial coils of different sizes
            A = max(coils_distance(i), coils_distance(j)) * 100; % Larger radius in
cm
            a = min(coils_distance(i), coils_distance(j)) * 100; % Smaller radius in
cm
            k_prim_square = ((A - a)^2 + d^2) / ((A + a)^2 + d^2);
            k_square = 1 - k_prim_square;

            if k_prim_square <= 0.1
                % Use table14
                base = abs(floor(log10(k_prim_square)));
                correction = base - abs(log10(k_prim_square));
                log_k_prim_square = base + correction;

                diff_values = log_k_prim_square -
log_k_prim_square_values_table14;
                valid_diff_indices = find(diff_values > 0);

                if ~isempty(valid_diff_indices)
                    [~, idx] = min(diff_values(valid_diff_indices));
                    idx = valid_diff_indices(idx);

                    if idx < length(log_k_prim_square_values_table14)
                        k1_log = log_k_prim_square_values_table14(idx);
                        k2_log = log_k_prim_square_values_table14(idx+1);
                        f1 = f_values_table14(idx);
                        f2 = f_values_table14(idx+1);

                        numStr = num2str(k1_log);
                        dotIndex = strfind(numStr, '.');
```

```
                    if isempty(dotIndex)
                        intPart = str2double(numStr);
                        fracPart = 0;
                    else
                        intPart = str2double(numStr(1:dotIndex-1));
                        fracPart = str2double(['0.'
numStr(dotIndex+1:end)]);
                    end
                    difference = -(intPart - fracPart);
                    k1 = 10^difference;

                    numStr = num2str(k2_log);
                    dotIndex = strfind(numStr, '.');

                    if isempty(dotIndex)
                        intPart = str2double(numStr);
                        fracPart = 0;
                    else
                        intPart = str2double(numStr(1:dotIndex-1));
                        fracPart = str2double(['0.'
numStr(dotIndex+1:end)]);
                    end
                    difference = -(intPart - fracPart);
                    k2 = 10^difference;

                    f = f1 + (k_prim_square - k1) * (f2 - f1) / abs(k2 - k1);
                    M_ij = f * sqrt(A * a) * 1e-6; % Mutual inductance in H
                end
            end

        elseif k_square <= 0.1
            % Use table15
            base = abs(floor(log10(k_square)));
            correction = base - abs(log10(k_square));
            log_k_square = base + correction;

            diff_values = log_k_square - log_k_square_values_table15;
```

```
                valid_diff_indices = find(diff_values > 0);

                if ~isempty(valid_diff_indices)
                    [~, idx] = min(diff_values(valid_diff_indices));
                    idx = valid_diff_indices(idx);

                    if idx < length(log_k_square_values_table15)
                        k1_log = log_k_square_values_table15(idx);
                        k2_log = log_k_square_values_table15(idx+1);
                        f1_log = log_f_values_table15(idx);
                        f2_log = log_f_values_table15(idx+1);

                        numStr = num2str(k1_log);
                        dotIndex = strfind(numStr, '.');

                        if isempty(dotIndex)
                            intPart = str2double(numStr);
                            fracPart = 0;
                        else
                            intPart = str2double(numStr(1:dotIndex-1));
                            fracPart = str2double(['0.'
numStr(dotIndex+1:end)]);
                        end
                        difference = -(intPart - fracPart);
                        k1 = 10^difference;

                        numStr = num2str(k2_log);
                        dotIndex = strfind(numStr, '.');

                        if isempty(dotIndex)
                            intPart = str2double(numStr);
                            fracPart = 0;
                        else
                            intPart = str2double(numStr(1:dotIndex-1));
                            fracPart = str2double(['0.'
numStr(dotIndex+1:end)]);
                        end
```

```
                                    difference = -(intPart - fracPart);
                                    k2 = 10^difference;

                                    numStr = num2str(f1_log);
                                    dotIndex = strfind(numStr, '.');
                                    intPart = str2double(numStr(1:dotIndex-1));
                                    fracPart = str2double(['0.' numStr(dotIndex+1:end)]);
                                    difference = -(intPart - fracPart);
                                    f1 = 10^difference;

                                    numStr = num2str(f2_log);
                                    dotIndex = strfind(numStr, '.');
                                    intPart = str2double(numStr(1:dotIndex-1));
                                    fracPart = str2double(['0.' numStr(dotIndex+1:end)]);
                                    difference = -(intPart - fracPart);
                                    f2 = 10^difference;

                                    f = f1 + (k_square - k1) * (f2 - f1) / (k2 - k1);
                                    M_ij = f * sqrt(A * a) * 1e-6; % Mutual inductance in H
                                end
                            end
                        else
                            % Use table13
                            f = interp1(k_prim_square_values_table13, f_values_table13,
k_prim_square, 'linear', 'extrap');
                            M_ij = f * sqrt(A * a) * 1e-6; % Mutual inductance in H
                        end
                    end

                    % Assign mutual inductance values
                    M(i, j) = M_ij;
                    M(j, i) = M_ij; % Symmetric matrix
                end
            end
        end

        return
```

```
writematrix(R, 'R_matrix.xlsx', 'Sheet', 1, 'Range', 'A1');
writematrix(M, 'M_matrix.xlsx', 'Sheet', 1, 'Range', 'A1');

disp('R and M matrices have been saved to Excel files.');
```

A.12 Eddy current calculation code

```
% The code is in Student_Thesis/2025_ypi/Code/eddy_current_thesis on the NAS.
% === Read resistance and inductance ===
M_origin = readmatrix('M_matrix.xlsx', 'Sheet', 1);
R_origin = readmatrix('R_matrix.xlsx', 'Sheet', 1);
R = R_origin;
M = M_origin;

total_num_coils = size(R, 1);

% === Read chamber data ===
chamber_data = readtable('chamber_components.xlsx');

% Extract the relevant data from the table
component_names = chamber_data.Component;                % Component names

% Initialize counters for inner and outer walls
num_inner_coils = 0;
num_outer_coils = 0;

% Initialize chamber components structures
chamber_components.inner_wall = struct();
chamber_components.outer_wall = struct();

% Iterate through each component and classify into inner or outer wall
for i = 1:height(chamber_data)
    component_name = component_names{i};

    if contains(component_name, 'Inner Wall')
        num_inner_coils = num_inner_coils + 1;

    elseif contains(component_name, 'Outer Wall')
        num_outer_coils = num_outer_coils + 1;

    end
end
```

```
fprintf('Number of inner wall coils: %d\n', num_inner_coils);
fprintf('Number of outer wall coils: %d\n', num_outer_coils);

% === Central solenoid current profile ===
N = 4000 + 1;              % Total number of time steps (2 segments of 2000 steps each)
t_end = 55e-3;             % Total duration (10 ms)
tspan = linspace(0, t_end, N); % Time array
I_max = 4000;              % A
dt = tspan(2) - tspan(1);          % Time step
I_profile = zeros(1,length(tspan));
t1 = 20e-3;
t2 = 20e-3;
t3 = 15e-3;

for i = 1:length(tspan)
    t = tspan(i);

    if t <= t1
        I_fun = @(t) (I_max / t1) * t;
        I_profile(i) = I_fun(t);

    elseif t <= t1+t2
        I_profile(i) = I_max;

    elseif t <= t1+t2+t3
        I_fun = @(t) (I_max / (t3)) * t;
        I_profile(i) = I_max - I_fun(t - (t1+t2));
    end
end

n_solenoid = 1;
V_history_solenoid = zeros(N, n_solenoid);
I_history_solenoid = zeros(N, n_solenoid);
I_prev_solenoid = zeros(1, n_solenoid);

M_solenoid = M(1,1);
R_solenoid = R(1,1);
```

```
var1 = (M_solenoid / dt);
for t = 2:length(tspan)
    I_solenoid = I_profile(t);
    V_solenoid = var1 * (I_solenoid - I_prev_solenoid) + R_solenoid * I_solenoid;
    V_history_solenoid(t,:) = V_solenoid;
    I_prev_solenoid = I_solenoid;
end

I_current = zeros(total_num_coils - 1, 1);    % Use -1 to exclude the central solenoid
I_prev_current = zeros(total_num_coils - 1, 1);
I_history_current = zeros(N, total_num_coils - 1);
inner_total_I_current = zeros(N, 1);
outer_total_I_current = zeros(N, 1);

M_voltage = M(2:end, 1);                % Mutual inductance between solenoid and walls
M_set = M(2:end, 2:end);                % Inductance matrix for walls
R_set = R(2:end, 2:end);                % Resistance matrix for walls

dI = (I_profile(2) - I_profile(1)) / dt;

tolerance = 1e-8;
max_iter = 1e10;

% Matrix for wall coil equations
A = (M_set / dt) + R_set;

for t = 2:length(tspan)

    % Inform every 1000 steps
    if mod(t, 1000) == 0
        fprintf('Processing time step %d out of %d\n', t, N);
    end

    % Voltage induced in wall coils
    V_current = M_voltage * dI;
```

```
        % Right-hand side of wall coil equations
        b = (M_set / dt) * I_prev_current - V_current;

        % Solve for wall coil currents
        [I_current, flag] = pcg(A, b, tolerance, max_iter, [], [], I_prev_current);

        % Store history
        I_history_current(t, :) = I_current;
        I_prev_current = I_current;

        % Sum eddy current
        inner_total_I_current(t) = sum(I_history_current(t, 1:num_inner_coils));
        outer_total_I_current(t) = sum(I_history_current(t,
num_inner_coils+1:num_inner_coils+num_outer_coils));

        dI = (I_profile(t) - I_profile(t - 1)) / dt;
end

figure;

yyaxis left;
plot(tspan*1000, inner_total_I_current, 'LineWidth', 2.5);
hold on;
plot(tspan*1000, outer_total_I_current, 'LineWidth', 2.5);
ylabel('Eddy Current (A)', 'FontWeight', 'bold');
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 2, 'FontWeight', 'bold');

yyaxis right;
plot(tspan*1000, I_profile/1000, 'LineWidth', 2.5);
ylabel('Current (A)', 'FontWeight', 'bold');

xlabel('Time (ms)', 'FontWeight', 'bold');
grid on;
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 1.5, 'FontWeight', 'bold');
legend('Inner wall', 'Outer wall','Solenoid', 'Location', 'southeast')

yyaxis left;
```

```
ylim([-max(abs(inner_total_I_current))*1.2 max(abs(inner_total_I_current))*1.2]);
yyaxis right;
ylim([-max(abs(I_profile/1000))*1.2 max(abs(I_profile/1000))*1.2]);

figure
plot(tspan*1000, I_history_current(:, 1:num_inner_coils), 'LineWidth', 2.5)
title(sprintf('N = %d', N))
grid on
xlabel('Time(ms)', 'FontWeight', 'bold');
ylabel('Current(A)', 'FontWeight', 'bold');
title('Eddy current of each inner vacuum-vessel wall')
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 1.5, 'FontWeight', 'bold');

plot(tspan*1000, I_history_current(:, num_inner_coils + 1:num_inner_coils +
num_outer_coils), 'LineWidth', 2.5)
title(sprintf('N = %d', N))
grid on
xlabel('Time(ms)', 'FontWeight', 'bold');
ylabel('Current(A)', 'FontWeight', 'bold');
title('Eddy current of each outer vacuum-vessel wall')
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 1.5, 'FontWeight', 'bold');
```

A.13 Loop voltage calculation code

```
% The code is in Student_Thesis/2025_ypi/Code/loop_V_eddy_thesis on the NAS.
% === Read data ===

% Read chamber data
chamber_data = readtable('chamber_components.xlsx');

% Extract the relevant data from the table
component_names = chamber_data.Component;          % Component names
r_values = chamber_data.Distance_m;                % Distance (r) values
z_values = chamber_data.Z_Position_m;              % Z-position values

% Initialize counters for inner and outer walls
num_inner_coils = 0;
num_outer_coils = 0;

% Initialize chamber components structures
chamber_components.inner_wall = struct();
chamber_components.outer_wall = struct();

% Iterate through each component and classify into inner or outer wall
for i = 1:height(chamber_data)
    component_name = component_names{i};

    if contains(component_name, 'Inner Wall')
        num_inner_coils = num_inner_coils + 1;

    elseif contains(component_name, 'Outer Wall')
        num_outer_coils = num_outer_coils + 1;
    end
end

% Display the results
fprintf('Number of inner wall coils: %d\n', num_inner_coils);
fprintf('Number of outer wall coils: %d\n', num_outer_coils);

all_coil_matrix = [r_values, z_values];
```

```
% Read solenoid parameters from the Excel file
solenoid_data = readtable('solenoid_parameters.xlsx');

% Extract the relevant parameters for the solenoid
solenoid_inner_num_coils = solenoid_data.Inner_Num_Coils;    % Number of inner coils
solenoid_outer_num_coils = solenoid_data.Outer_Num_Coils;    % Number of outer coils
solenoid_height = solenoid_data.Total_Height_m;                    % Total height of
solenoid
solenoid_inner_distance = solenoid_data.Inner_Distance_m;    % Inner coil distance from
center
solenoid_outer_distance = solenoid_data.Outer_Distance_m;    % Outer coil distance from
center
solenoid_num_coils = solenoid_inner_num_coils * 2;

% === CS current profile and eddy current in inner and outer walls ===

% Read inductance and resistance matrices
R0 = readmatrix('R_matrix.xlsx', 'Sheet', 1);
M0 = readmatrix('M_matrix.xlsx', 'Sheet', 1);
M = M0;
R = R0;

total_num_coils = size(R, 1);

% Define solenoid parameters
V_loop_desired = 0.066; % Desired loop voltage (V)
mu0 = 4*pi*1e-7; % Vacuum permeability (H/m)
solenoid_major_radius = (solenoid_inner_distance + solenoid_outer_distance)/2;
% Radius of the solenoid [m]
A_solenoid = pi * solenoid_major_radius^2;

% Calculate current ramp rate to achieve desired loop voltage
I_rate = - (V_loop_desired * solenoid_height) / (A_solenoid * mu0 * solenoid_num_coils);
% Current ramp rate (A/s)

% Time parameters
```

```
N_steps = 500 + 1; % Number of time steps
t_end = 15e-3; % Total time (s)
tspan = linspace(0, t_end, N_steps); % Time array
dt = tspan(2) - tspan(1); % Time step (s)

% Define current profile (linear ramp)
I_profile = abs(t_end * I_rate) + (tspan * I_rate);

n_solenoid = 1;

V_history_solenoid = zeros(N_steps, n_solenoid);
I_history_solenoid = zeros(N_steps, n_solenoid);
I_prev_solenoid = zeros(1, n_solenoid);

M_solenoid = M(1,1);
R_solenoid = R(1,1);

var1 = (M_solenoid / dt);
for t = 2:length(tspan)
    I_solenoid = I_profile(t);
    V_solenoid = var1 * (I_solenoid - I_prev_solenoid) + R_solenoid * I_solenoid;
    V_history_solenoid(t,:) = V_solenoid;
    I_prev_solenoid = I_solenoid;
end

I_current = zeros(total_num_coils - 1 , 1);    % Use -1 to exclude the central solenoid
I_prev_current = zeros(total_num_coils - 1    , 1);
I_history_current = zeros(N_steps, total_num_coils -1    );
inner_total_I_current = zeros(N_steps, 1);
outer_total_I_current = zeros(N_steps, 1);

M_voltage = M(2:end, 1);                  % Mutual inductance between solenoid and walls
M_set = M(2:end, 2:end);                  % Inductance matrix for walls
R_set = R(2:end, 2:end);                  % Resistance matrix for walls

dI = (I_profile(2) - I_profile(1)) / dt;
```

```
tolerance = 1e-8;
max_iter = 1e10;

% Matrix for wall coil equations
A = (M_set / dt) + R_set;

for t = 2:length(tspan)

    % Voltage induced in wall coils
    V_current = M_voltage * dI;

    % Right-hand side of wall coil equations
    b = (M_set / dt) * I_prev_current - V_current;

    % Solve for wall coil currents
    [I_current, flag] = pcg(A, b, tolerance, max_iter, [], [], I_prev_current);

    % Store history
    I_history_current(t, :) = I_current;
    I_prev_current = I_current;

    % Sum eddy current
    inner_total_I_current(t) = sum(I_history_current(t, 1:num_inner_coils));
    outer_total_I_current(t) = sum(I_history_current(t,
num_inner_coils+1:num_inner_coils+num_outer_coils));

    dI = (I_profile(t) - I_profile(t - 1)) / dt;
end

figure;
set(gcf, 'Renderer', 'painters');

yyaxis left
plot(tspan * 1e3, I_profile / 1e3, 'LineWidth', 2.5);    % kA
ylabel('Current (kA)');

leftMax = max(I_profile) / 1e3;
```

```
ylim([0, leftMax * 1.1]);

yyaxis right
plot(tspan * 1e3, inner_total_I_current, 'LineWidth', 2.5); hold on;
plot(tspan * 1e3, outer_total_I_current, 'LineWidth', 2.5);
ylabel('Eddy Current (A)');

rightMax = max([max(inner_total_I_current), max(outer_total_I_current)]);
ylim([0, rightMax * 1.5]);

xlabel('Time (ms)');
title(['Current profile for V_{loop} = ', num2str(V_loop_desired, '%.3f')]);
legend('Central Solenoid', 'Inner wall', 'Outer wall', 'Location', 'north');
grid on;
box on;
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 1.5, 'FontWeight', 'bold');

figure
plot(tspan*1000, I_history_current(:, 1:num_inner_coils), 'LineWidth', 2.5)
grid on
xlabel('Time (ms)', 'FontWeight', 'bold');
ylabel('Current (A)', 'FontWeight', 'bold');
title('Eddy current of each inner chamber wall')
set(gca, 'LineWidth', 1.5, 'FontSize', 14, 'FontWeight', 'bold');
max_inner = max(I_history_current(:, 1:num_inner_coils), [], 'all');
ylim([0, 1.2 * max_inner]);

figure
plot(tspan*1000, I_history_current(:, 1+num_inner_coils :
num_inner_coils+num_outer_coils), 'LineWidth', 2.5)
grid on
xlabel('Time (ms)', 'FontWeight', 'bold');
ylabel('Current (A)', 'FontWeight', 'bold');
title('Eddy current of each outer chamber wall')
set(gca, 'LineWidth', 1.5, 'FontSize', 14, 'FontWeight', 'bold');
max_outer = max(I_history_current(:, 1+num_inner_coils :
num_inner_coils+num_outer_coils), [], 'all');
```

```
ylim([0, 1.2 * max_outer]);

% === CS Bz Calculation ===

num_slices = 100;
spatial_length = 1001;
spatial_grid = 1;
dz = solenoid_height / num_slices;
r_values = linspace(0, spatial_grid, spatial_length);
z_obs = 0;
y_obs = 0;
Nx_obs = spatial_length;
BZ_x_solenoid = zeros(N_steps, Nx_obs);

% Compute the magnetic flux induced by the central solenoid (CS) only, and then obtain
V_loop_time using the gradient
Phi_solenoid = zeros(N_steps, 1);
R_loop = 0.085; % Expected location of plasma breakdown

for t_idx = 1:length(tspan)
    I = I_profile(t_idx);
    Bz_total_r = zeros(size(r_values));

    for idx = 1:length(r_values)
        x_obs = r_values(idx);
        Bz_total = 0;

        % Accumulate Bz from each loop
        for n = 1:num_slices
            z0 = -solenoid_height/2 + (n - 0.5) * dz;
            I_loop = I * solenoid_num_coils / num_slices;
            [~, ~, Bz] = magnetic_field_loop(solenoid_major_radius, I_loop, x_obs,
y_obs, z_obs - z0);
            Bz_total = Bz_total + Bz;
        end

        Bz_total_r(idx) = Bz_total;
```

```
        end

        BZ_x_solenoid(t_idx,:) = Bz_total_r;

        % Calculate magnetic flux
        indices = r_values <= R_loop;
        r_in_loop_local = r_values(indices);
        Bz_in_loop_local = Bz_total_r(indices);
        Phi_solenoid(t_idx) = trapz(r_in_loop_local, Bz_in_loop_local .* 2 .* pi .*
r_in_loop_local);
end

% Compute the loop voltage from the central solenoid using the gradient
V_loop_time = -gradient(Phi_solenoid, dt);

%% Bz generated by the solenoid vs radius
figure;
set(gcf, 'Renderer', 'painters');

plot(r_values*1000, BZ_x_solenoid(100, :), 'LineWidth', 2.5);
xlabel('X Position (mm)');
ylabel('Magnetic Field B_z (T)');
title('Magnetic Field B_z vs Radius');
grid on;
ylim([-0.1 2.6]);
xlim([0 200]);
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 2, 'FontWeight', 'bold');
print(gcf, 'cs_Bz_vs_radius', '-dpng', '-r600');

figure;
plot(tspan*1000, V_loop_time, 'LineWidth', 2.5);
xlabel('time (ms)'); ylabel('V_{loop} (V)');
title('V_{loop} from only central solenoid');
grid on;
set(gca, 'linewidth', 1.5, 'fontsize', 12, 'FontWeight', 'bold');

% === Inner and Outer Chamber Wall Bz Calculation ===
```

```
total_num_wall = length(all_coil_matrix);

N_segments_wall    = 30;
u0 = 4*pi*1e-7;
xp = linspace(0, spatial_grid, Nx_obs);

num_wall_coils = total_num_wall;
spec_matrix = zeros(num_wall_coils, Nx_obs);

fprintf('Starting precomputation of spec_matrix...\n');
for coil = 1:num_wall_coils
    R = all_coil_matrix(coil, 1);
    z0 = all_coil_matrix(coil, 2);

    phi = linspace(-pi/2, 3*pi/2, N_segments_wall);
    Xc = R * cos(phi);
    Yc = R * sin(phi);
    Zc = z0 * ones(size(Xc));

    X_next = circshift(Xc, -1);
    Y_next = circshift(Yc, -1);
    Z_next = circshift(Zc, -1);

    dlx = X_next - Xc;
    dly = Y_next - Yc;
    dlz = Z_next - Zc;

    x_mid = 0.5 * (Xc + X_next);
    y_mid = 0.5 * (Yc + Y_next);
    z_mid = 0.5 * (Zc + Z_next);

    for a = 1:Nx_obs
        x_obs = xp(a);
        y_obs = 0;
        z_obs = 0;
```

```
            Rx = x_obs - x_mid;
            Ry = y_obs - y_mid;
            Rz = z_obs - z_mid;

            Xcross = dly .* Rz - dlz .* Ry;
            Ycross = dlz .* Rx - dlx .* Rz;
            Zcross = dlx .* Ry - dly .* Rx;

            R_dist = sqrt(Rx.^2 + Ry.^2 + Rz.^2);
            valid = R_dist ~= 0;
            Bz = zeros(size(R_dist));
            Bz(valid) = (1 * u0) / (4*pi) .* Zcross(valid) ./ (R_dist(valid).^3);

            spec_matrix(coil, a) = sum(Bz);
        end

        if mod(coil, 100) == 0
            fprintf('Completed precomputing coil %d / %d\n', coil, num_wall_coils);
        end
    end
    fprintf('Completed precomputation of spec_matrix.\n');

    I_all_wall_matrix = I_history_current; % [time x coils]
    BZ_x_wall = I_all_wall_matrix * spec_matrix; % [time x Nx_obs]

    I_all_wall_reshaped = reshape(I_all_wall_matrix, [N_steps, num_wall_coils, 1]);
    spec_matrix_reshaped = reshape(spec_matrix, [1, num_wall_coils, Nx_obs]);
    BZ_x_coils_wall = I_all_wall_reshaped .* spec_matrix_reshaped;

    % === Calculate total loop voltage from CS and Eddy current ===

    Bz_total = BZ_x_wall + BZ_x_solenoid;

    indices = r_values <= R_loop;
    r_in_loop = r_values(indices);
    Bz_in_loop = Bz_total(:, indices);
```

```
Phi = trapz(r_in_loop, Bz_in_loop .* 2 .* pi .* r_in_loop, 2);
V_loop_total = -gradient(Phi, dt);

% === Plot loop voltage comparison ===
figure;
set(gcf, 'Renderer', 'painters');

plot(tspan*1000, V_loop_total, 'LineWidth', 2.5);
hold on;
plot(tspan*1000, V_loop_time, 'LineWidth', 2.5);
xlabel('Time (ms)');
ylabel('V_{loop} (V)');
title('V_{loop} induced by CS and Eddy current');
legend('CS and Eddy current', 'Only CS', 'Location', 'southeast');
grid on;
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 2, 'FontWeight', 'bold');

figure;
set(gcf, 'Renderer', 'painters');

V_loop_total_percentage = (V_loop_total ./ V_loop_time) * 100;
V_loop_time_percentage = (V_loop_time ./ V_loop_time) * 100;

plot(tspan*1000, V_loop_total_percentage, 'LineWidth', 2.5);
hold on;
plot(tspan*1000, V_loop_time_percentage, 'LineWidth', 2.5);

xlabel('Time (ms)');
ylabel('V_{loop} (%)');
ylim([40 110]);
title('V_{loop} induced by CS and Eddy current');
legend('CS and Eddy current', 'Only CS (100%)', 'Location', 'southeast');
grid on;
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 2, 'FontWeight', 'bold');

% === magnetic_field_loop Function Definition ===
function [Bx, By, Bz] = magnetic_field_loop(R, I, x, y, z)
```

```
    mu0 = 4*pi*1e-7;
    rho = sqrt(x.^2 + y.^2);
    phi = atan2(y, x);

    rho(rho == 0) = eps;

    k_squared = 4 * R * rho ./ ((R + rho).^2 + z.^2);
    k = sqrt(k_squared);
    [K, E] = ellipke(k_squared);

    Brho = (mu0 * I) ./ (2 * pi * sqrt((R + rho).^2 + z.^2)) .* (z ./ rho) .* ...
            ( (R^2 + rho.^2 + z.^2) ./ ((R - rho).^2 + z.^2) .* E - K );
    Bz = (mu0 * I) ./ (2 * pi * sqrt((R + rho).^2 + z.^2)) .* ...
        ( (R^2 - rho.^2 - z.^2) ./ ((R - rho).^2 + z.^2) .* E + K );

    Bx = Brho .* cos(phi);
    By = Brho .* sin(phi);
end
```

A.14 Plasma parameters calculation code

```
% The code is in Student_Thesis/2025_ypi/Code/plasma_thesis on the NAS.
% === Read data ===

% Plasma
plasma_data = readtable('plasma_parameters.xlsx');

% Assign the values back to their original variable names
a_plasma = plasma_data.Semi_Minor_Axis_m(1);                    % Plasma cross-section semi-minor axis (m)
kappa = plasma_data.Kappa(1);                    % Plasma cross-section semi-major axis (m)
b_plasma = a_plasma * kappa;
R_plasma = plasma_data.Major_Radius_m(1);                    % Major radius (m)
A_plasma = plasma_data.Cross_Sectional_Area_m2(1);                    % Plasma cross-sectional area (m^2)
L_plasma = plasma_data.Length_m(1);
V_plasma = plasma_data.Volume_m3(1);                    % Plasma volume (m^3) for toroidal geometry

plasma_material = 'He';            % H, He, Ar
n_total_plasma = 1e17;              % Total particle density (m^-3)
readRateData(plasma_material);
B0    = 0.1;                        % Central magnetic field (T)
T_plasma = 0.026; % Initial plasma temperature in eV (300K)
ne = 0;
n0 = 0;
eV_to_J = 1.60218e-19;    % Conversion factor from eV to J

% Read chamber data
chamber_data = readtable('chamber_components.xlsx');

% Extract the relevant data from the table
component_names = chamber_data.Component;                % Component names

% Initialize counters for inner and outer walls
num_inner_coils = 0;
```

```
num_outer_coils = 0;

% Initialize chamber components structures
chamber_components.inner_wall = struct();
chamber_components.outer_wall = struct();

% Iterate through each component and classify into inner or outer wall
for i = 1:height(chamber_data)
    component_name = component_names{i};

    if contains(component_name, 'Inner Wall')
        num_inner_coils = num_inner_coils + 1;

    elseif contains(component_name, 'Outer Wall')
        num_outer_coils = num_outer_coils + 1;

    end
end

% Display the results
fprintf('Number of inner wall coils: %d\n', num_inner_coils);
fprintf('Number of outer wall coils: %d\n', num_outer_coils);

% === Calculation ===

% Time range and current profile
n_solenoid = 1;
N = 2000 + 1;
N_q = 100;
t1 = 10e-3;          % First segment
t2 = 40e-3;          % Second segment
t_total = t1 + t2;
tspan = linspace(0, t_total, N); % Time array
dt = tspan(2) - tspan(1);          % Time step
I1 = 10e3;           % Maximum current in the first segment
I2 = 7.5e3;          % Maximum current in the second segment
```

```
% Define the current profile
I_profile = zeros(size(tspan));
for i = 1:length(tspan)
    t = tspan(i);
    if t <= t1
        % First phase: From I1 kA to 0 A
        I_profile(i) = I1 * (1 - t / t1);
    elseif t <= t1 + t2
        % Second phase: From 0 A to -I2 kA
        I_profile(i) = -I2 * ((t - t1) / t2);
    end
end

M_origin = readmatrix('M_matrix.xlsx', 'Sheet', 1);
R_origin = readmatrix('R_matrix.xlsx', 'Sheet', 1);
M = M_origin;
R = R_origin;
total_num_coils = size(R, 1);

% Initialization
V_history_solenoid = zeros(N, n_solenoid);
I_history_solenoid = zeros(N, n_solenoid);
I_prev_solenoid = zeros(1, n_solenoid);

M_solenoid = M(1, 1);
R_solenoid = R(1, 1);
var1 = (M_solenoid / dt);

% Plasma parameters to track over time
T_eV_array = zeros(1, N);
gamma_array = zeros(1, N);
ne_array = zeros(1, N);
n0_array = zeros(1, N);
P_ohmic_array = zeros(1, N);
P_prb_array = zeros(1, N);
P_ion_array = zeros(1, N);
P_line_array = zeros(1, N);
```

```
P_loss_array = zeros(1, N);
P_net_array = zeros(1, N);
Rp_array = zeros(1, N);
q_array = zeros(1, N);

I_current = zeros(total_num_coils - 1, 1);
I_prev_current = zeros(total_num_coils - 1, 1);
I_history_current = zeros(N, total_num_coils - 1);
inner_total_I_current = zeros(N, 1);
outer_total_I_current = zeros(N, 1);
plasma_I_current = zeros(N,1);

% Energy tracking
U_array = zeros(1, N);                  % Total thermal energy (J)
E_input_array = zeros(1, N);        % Cumulative input energy (J)
E_loss_array = zeros(1, N);        % Cumulative loss energy (J)

[Rp, eta_N, T_next, P_ohmic, P_prb, P_ion, P_line, P_loss, P_net,...
      ne, n0, gamma, Lp] = plasma_parameters(0, T_plasma, dt, ne, n0, ...
      n_total_plasma, a_plasma, kappa, R_plasma, plasma_material, B0, N_q);

R(total_num_coils, total_num_coils) = Rp;
M(total_num_coils, total_num_coils) = Lp;
M_set = M(2:end, 2:end);                % Inductance matrix for walls and plasma
R_set = R(2:end, 2:end);                % Resistance matrix for walls and plasma
M_voltage = M(2:end, 1);                % Mutual inductance between solenoid and walls
and plasma
A = (M_set / dt) + R_set;

sigma_neo_array = zeros(1, N);
ft_array = zeros(1, N);
nuestar_array = zeros(1, N);
q_profile_array = zeros(N_q, N);

tolerance = 1e-8;
max_iter = 1e10;
```

```
for t = 1:N
    if mod(t, 100) == 0
        fprintf('Processing time step %d out of %d\n', t, N);
    end

    if t == 1
        dI = 0;
    else
        dI = (I_profile(t) - I_profile(t - 1)) / dt;
    end

    V_current = M_voltage * dI;
    b = (M_set / dt) * I_prev_current - V_current;
    [I_current, flag] = pcg(A, b, tolerance, max_iter, [], [], I_prev_current);

    I_history_current(t, :) = I_current;
    I_prev_current = I_current;

    inner_total_I_current(t) = sum(I_current(1:num_inner_coils));
    outer_total_I_current(t) =
sum(I_current(num_inner_coils+1:num_inner_coils+num_outer_coils));
    plasma_I_current(t) = I_current(end);

    [Rp, eta_N, T_next, P_ohmic, P_prb, P_ion, P_line, P_loss, P_net,...
      ne, n0, gamma, Lp, q, sigma_neo, ft, nuestar, q_vals] = ...
    plasma_parameters(plasma_I_current(t), T_plasma, dt, ne, n0, ...
        n_total_plasma, a_plasma, kappa, R_plasma, plasma_material, B0, N_q);

    T_plasma = T_next;
    R_set(end, end) = Rp; % update plasma resistance
    A = (M_set / dt) + R_set;

    T_eV_array(t) = T_next;
    gamma_array(t) = gamma;
    ne_array(t) = ne;
    n0_array(t) = n0;
    P_ohmic_array(t) = P_ohmic;
```

```
        P_prb_array(t) = P_prb;
        P_ion_array(t) = P_ion;
        P_line_array(t) = P_line;
        P_loss_array(t) = P_loss;
        P_net_array(t) = P_net;
        Rp_array(t) = Rp;
        q_array(t) = q;
        sigma_neo_array(t) = sigma_neo;
        ft_array(t) = ft;
        nuestar_array(t) = nuestar;
        q_profile_array(:, t) = q_vals;

        U_array(t) = 1.5 * n_total_plasma * V_plasma * T_plasma * eV_to_J;
        if t == 1
            E_input_array(t) = P_ohmic * dt * V_plasma;
            E_loss_array(t) = P_loss * dt * V_plasma;
        else
            E_input_array(t) = E_input_array(t-1) + P_ohmic * dt * V_plasma;
            E_loss_array(t) = E_loss_array(t-1) + P_loss * dt * V_plasma;
        end
end

% Eddy current of inner and outer chamber
figure
plot(tspan*1000, inner_total_I_current/1000, 'LineWidth', 2.5);
hold on;
plot(tspan*1000, outer_total_I_current/1000, 'LineWidth', 2.5);
grid on;
xlabel('Time(ms)', 'FontWeight', 'bold');
ylabel('Current (kA)', 'FontWeight', 'bold');
title('Eddy current')
legend('Inner wall', 'Outer wall', 'Location', 'southeast')
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 2, 'FontWeight', 'bold');

figure;

yyaxis left
```

```
plot(tspan * 1e3, I_profile / 1e3, 'LineWidth', 2.5);    % ms vs. kA
ylabel('Central Solenoid Current (kA)');

leftMax = max(I_profile) / 1e3;                % kA
leftMin = min(I_profile) / 1e3;

ylim([leftMin, leftMax * 1.1]);

yyaxis right
plot(tspan * 1e3, plasma_I_current / 1e3, 'LineWidth', 2.5); hold on;
ylabel('Current (kA)');

rightMax = max([max(plasma_I_current)]) / 1e3;
ylim([0, rightMax * 1.1]);

xlabel('Time (ms)');
title('Current Profile');
legend('Central Solenoid', 'Plasma current', 'Location', 'best');
grid on;
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 2, 'FontWeight', 'bold');

% Temperature vs. Time
figure;
plot(tspan*1e3, T_eV_array, 'LineWidth', 2.5);
xlabel('Time (ms)');
ylabel('Temperature (eV)');
title('Plasma Temperature');
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 2, 'FontWeight', 'bold');
grid on;

% Ionization Fraction vs. Time
figure;
plot(tspan*1e3, gamma_array, 'LineWidth', 2.5);
xlabel('Time (ms)');
ylabel('Ionization Fraction \gamma');
title('Ionization Fraction');
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 2, 'FontWeight', 'bold');
```

```
grid on;

% Plasma Resistance vs. Time
plot(tspan * 1e3, Rp_array, 'LineWidth', 2.5);
xlabel('Time (ms)');
ylabel('Resistance R_p (\Omega)');
title('Plasma Resistance');
set(gca, 'YScale', 'log');
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 2, 'FontWeight', 'bold');
set(gca, 'YMinorGrid','off')
grid on

% Ohmic Heating Power Density vs. Time
figure;
plot(tspan*1e3, P_ohmic_array, 'LineWidth', 2.5);
xlabel('Time (ms)');
ylabel('P_{ohmic} (W/m^3)');
title('Ohmic Heating Power Density');
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 2, 'FontWeight', 'bold');
grid on;

% Plot Multiple Power Losses on the Same Figure
figure;
hold on;

plot(tspan*1000, P_prb_array, 'LineWidth', 2.5, 'DisplayName', 'Bremsstrahlung
(P_{prb})');
plot(tspan*1000, P_line_array, 'LineWidth', 2.5, 'DisplayName', 'Line Radiation
(P_{line})');
plot(tspan*1000, P_ion_array, 'LineWidth', 2.5, 'DisplayName', 'Ionization Loss
(P_{ion})');
plot(tspan*1000, P_loss_array, 'LineWidth', 2.5, 'DisplayName', 'Total Loss (P_{loss})');

xlabel('Time (ms)', 'FontWeight', 'bold');
ylabel('Power Density (W/m^3)', 'FontWeight', 'bold');
legend('Location', 'best');
title('Energy loss');
```

```
grid on;
box on
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 2, 'FontWeight', 'bold');

% Plot Total Power Loss
figure
plot(tspan*1000, P_net_array, 'LineWidth', 2.5);
hold on;
grid on;
xlabel('Time (ms)', 'FontWeight', 'bold');
ylabel('Power Density (W/m^3)', 'FontWeight', 'bold');
title('Power Input');
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 2, 'FontWeight', 'bold');

%% Energy Conservation Check
figure;
plot(tspan * 1e3, E_input_array, 'LineWidth', 2.5);
hold on;
plot(tspan * 1e3, E_loss_array, 'LineWidth', 2.5);
plot(tspan * 1e3, U_array - U_array(1), 'LineWidth', 2.5);
plot(tspan * 1e3, E_loss_array + U_array - U_array(1), '--', 'LineWidth', 2.5);
xlabel('Time (ms)', 'FontWeight', 'bold');
ylabel('Energy (J)', 'FontWeight', 'bold');
title('Energy Conservation Check');
legend('Ohmic heating (Input Energy)', 'Total loss', 'Thermal Energy', 'Output Energy',
'Location', 'best');
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 2, 'FontWeight', 'bold');
grid on;

function [Rp, eta_N, T_next, P_ohmic, P_prb, P_ion, P_line, P_loss, P_net, ...
            ne, n0, gamma, Lp, q, sigma_neo, ft, nuestar, q_vals] = ...
            plasma_parameters(I, T_plasma, dt, ne, n0, ...
                                    n_total_plasma, a_plasma, kappa, R_plasma,
plasma_material, B0, N_q)

    %% === Constant Definitions ===
    e     = 1.6e-19;                % Elementary charge (C)
```

```
kB    = 1.38e-23;               % Boltzmann constant (J/K)
mu0 = 4 * pi * 1e-7;        % Permeability of vacuum (H/m)
T_e = T_plasma;               % (eV)
eV_to_K = 11604.52;         % Conversion factor from eV to K

switch plasma_material
case 'H'
    Z1_use = 1;
    Ei_eV = 13.6;
    mi = 1.67e-27;
    sigma_ei = 1.5e-16 * T_e^(-2);
    sigma_ea = 3e-19      * T_e^(-0.5);

case 'He'
    Ei_eV = 24.6;
    mi = 6.64e-27;
    Z1_use = 1;    % He⁺ → He⁰ dominates
    sigma_ei = 1.5e-16 * T_e^(-2);
    sigma_ea = 1e-19      * T_e^(-0.46);

case 'Ar'
    Ei_eV = 15.76;
    mi = 6.63e-26;
    Z1_use = 1;
    sigma_ei = 3.0e-16 * T_e^(-2);
    sigma_ea = 2.0e-19 * T_e^(-0.5);

otherwise
    error('Unknown plasma material: %s', plasma_material);
end

Ei_J = Ei_eV * e;

%% === Geometric Parameters ===
b_plasma = a_plasma * kappa;              % Plasma minor radius a_plasma, major radius b_plasma
A_plasma = pi * a_plasma * b_plasma; % Plasma cross-sectional area (m^2)
```

```
L_plasma = 2 * pi * R_plasma;                % Effective current path length (m)
V_plasma = 2 * pi^2 * R_plasma * a_plasma * b_plasma; % Plasma volume (m^3)

% Calculation of q profile
r_vals = linspace(1e-4, a_plasma, N_q);
q_vals = zeros(1, N_q);
for i = 1:N_q
    r = r_vals(i);
    J_p = I / (pi*r^2);
    B_T = B0 * R_plasma / (R_plasma + r);
    B_P = mu0 * J_p / (2*pi*max(r,1e-3));
    q_vals(i) = (r * B_T) / ((R_plasma + r)* B_P);
end
q = sum(q_vals .* r_vals) / sum(r_vals); % scalar q output

gamma_collision = sigma_ea / sigma_ei;
gamma_collision = max(min(gamma_collision, 1), 0);

% SCD/ACD rates
R_ion = get_scd_rate(T_e, ne, Z1_use);    % (cm^3/s)
R_rec = get_acd_rate(T_e, ne, Z1_use);    % (cm^3/s)
if isempty(R_ion), R_ion = 0; end
if isempty(R_rec), R_rec = 0; end

R_ion_m3 = R_ion * 1e-6;      % (m^3/s)
R_rec_m3 = R_rec * 1e-6;      % (m^3/s)

dne_dt    = R_ion_m3 * (n0 * ne) - R_rec_m3 * (ne^2);
ne_scd    = ne + dne_dt * dt;
ne_scd    = max(ne_scd, 0);
gamma_scd = ne_scd / n_total_plasma;
gamma_scd = max(min(gamma_scd, 1), 0);

slope = 10;
x_start = 0.0;
x_end = 0.5;
```

```
x_norm = (gamma_collision - x_start) / (x_end - x_start);
x_norm = min(max(x_norm, 0), 1);

w_scd = 1 ./ (1 + exp(-slope * (x_norm - x_end)));

w_collision = 1 - w_scd;
gamma = w_collision .* gamma_collision + w_scd .* gamma_scd;

if gamma > 0.9999
gamma = 1;
end

ne = gamma * n_total_plasma;
n0 = (1 - gamma) * n_total_plasma;

%% === Neoclassical Conductivity Calculation ===
ne_for_neo = ne;
te_for_neo = T_e;
ni_for_neo = ne_for_neo;
ti_for_neo = te_for_neo;
Zeff = 1.0;
eps_local = a_plasma / R_plasma;

[nuestar, nuistar] = nustar(ne_for_neo, te_for_neo, ni_for_neo, ti_for_neo, Zeff, q,
R_plasma, eps_local);
eps = a_plasma / R_plasma;
delta = 0;
ft = ftav_with_delta(eps, delta);

[sigma_neo, ~] = sigmaneo(ft, ne, T_e, Zeff, nuestar);
eta_N = 1 ./ sigma_neo;
Rp = eta_N * (L_plasma / A_plasma);

%% === Ohmic Heating Power Calculation ===
P_ohmic = (I^2 * Rp) / V_plasma;

%% Bremsstrahlung, Ionization Loss, Line Radiation ===
```

```
        R_prb = get_prb_rate(T_e, ne, Z1_use);

        if isempty(R_prb), R_prb = 0; end
        R_prb_m3 = R_prb * 1e-6;
        P_prb = R_prb_m3 * (ne^2);

        P_ion = R_ion_m3 * ne * n0 * Ei_J;

        R_plt = get_plt_rate(T_e, ne, Z1_use);

        if isempty(R_plt), R_plt = 0; end
        R_plt_m3 = R_plt * 1e-6;
        P_line = R_plt_m3 * (ne^2);

        P_loss = P_prb + P_ion + P_line;
        P_net  = P_ohmic - P_loss;

        %% === Heat Capacity Calculation & Temperature Update ===
        c_v      = n_total_plasma * (1.5 * kB);   % J/K·m³
        dT_dt    = P_net / c_v;                   % K/s
        dT_dt_eV = dT_dt / eV_to_K;               % eV/s
        T_next   = T_plasma + dT_dt_eV * dt;      % eV

        T_next = max(min(T_next, 1e8 / eV_to_K), 300 / eV_to_K);

        %% Plasma Inductance Calculation ===
        li_plasma = 0.5;
        Lp = mu0 * R_plasma * (log(8 * R_plasma / a_plasma) + li_plasma/2 - 2);

end

% === Read ADAS data ===

function readRateData(material)
        material = lower(material);
        global rateMatrix_scd Te_scd Ne_scd
        global rateMatrix_acd Te_acd Ne_acd
```

```
global rateMatrix_prb Te_prb Ne_prb
global rateMatrix_plt Te_plt Ne_plt

dataFolder = 'ADAS';
types = {'scd', 'acd', 'prb', 'plt'};

for i = 1:length(types)
    type = types{i};
    fname = fullfile(dataFolder, [type '_' material '.dat']);
    if ~isfile(fname), warning('%s file not found', fname); continue; end
    lines = readlines(fname, "EmptyLineRule","skip");

    meta = sscanf(lines(1), '%f');
    num_Z1 = meta(1);
    num_Ne = meta(2);
    num_Te = meta(3);
    grid_header_lines = ceil(num_Ne / 8);
    temp_header_lines = ceil(num_Te / 8);

    % Read ne, Te
    logNe = [];
    for j = 3:(2 + grid_header_lines)
        logNe = [logNe; sscanf(lines(j), '%f')];
    end
    logTe = [];
    start_te = 3 + grid_header_lines;
    for j = start_te:(start_te + temp_header_lines - 1)
        logTe = [logTe; sscanf(lines(j), '%f')];
    end
    Ne = 10.^logNe;
    Te = 10.^logTe;

    idx_z1 = find(contains(lines, '/ Z1='));
    idx_z1 = [idx_z1; length(lines) + 1];

    rateStruct = struct();
    for z = 1:num_Z1
```

```
            startLine = idx_z1(z) + 1;
            endLine = idx_z1(z + 1) - 1;
            blockLines = lines(startLine:endLine);
            logRate = [];
            for k = 1:length(blockLines)
                logRate = [logRate; sscanf(blockLines(k), '%f')];
            end
            if numel(logRate) ~= num_Te * num_Ne
                warning("Z1=%d rate matrix size mismatch: expected %d, got %d", ...
                    z, num_Te * num_Ne, numel(logRate));
                continue;
            end
            mat = reshape(logRate, [num_Te, num_Ne]);
            rateStruct.(['Z1_', num2str(z)]) = 10.^mat;
        end

        % Save by type
        switch type
            case 'scd'
                Te_scd = Te; Ne_scd = Ne; rateMatrix_scd = rateStruct;
            case 'acd'
                Te_acd = Te; Ne_acd = Ne; rateMatrix_acd = rateStruct;
            case 'prb'
                Te_prb = Te; Ne_prb = Ne; rateMatrix_prb = rateStruct;
            case 'plt'
                Te_plt = Te; Ne_plt = Ne; rateMatrix_plt = rateStruct;
        end
    end
end

function R = get_scd_rate(T_e, ne, Z1)
    global Te_scd Ne_scd rateMatrix_scd
    key = ['Z1_', num2str(Z1)];
    if ~isfield(rateMatrix_scd, key)
        warning('No data for Z1 = %d in SCD', Z1); R = 0; return;
    end
    ne_cm3 = ne / 1e6;
```

```
    R = interp2(Ne_scd(:)', Te_scd(:), rateMatrix_scd.(key), ne_cm3, T_e, 'linear', 0);
end

function R = get_acd_rate(T_e, ne, Z1)
    global Te_acd Ne_acd rateMatrix_acd
    key = ['Z1_', num2str(Z1)];
    if ~isfield(rateMatrix_acd, key)
        warning('No data for Z1 = %d in ACD', Z1); R = 0; return;
    end
    ne_cm3 = ne / 1e6;
    R = interp2(Ne_acd(:)', Te_acd(:), rateMatrix_acd.(key), ne_cm3, T_e, 'linear', 0);
end

function R = get_prb_rate(T_e, ne, Z1)
    global Te_prb Ne_prb rateMatrix_prb
    key = ['Z1_', num2str(Z1)];
    if ~isfield(rateMatrix_prb, key)
        warning('No data for Z1 = %d in PRB', Z1); R = 0; return;
    end
    ne_cm3 = ne / 1e6;
    R = interp2(Ne_prb(:)', Te_prb(:), rateMatrix_prb.(key), ne_cm3, T_e, 'linear', 0);
end

function R = get_plt_rate(T_e, ne, Z1)
    global Te_plt Ne_plt rateMatrix_plt
    key = ['Z1_', num2str(Z1)];
    if ~isfield(rateMatrix_plt, key)
        warning('No data for Z1 = %d in PLT', Z1); R = 0; return;
    end
    ne_cm3 = ne / 1e6;
    R = interp2(Ne_plt(:)', Te_plt(:), rateMatrix_plt.(key), ne_cm3, T_e, 'linear', 0);
end

% === Reference: https://gitlab.epfl.ch/spc/public/NEOS. ===

function [nuestar, nuistar] = nustar(ne,te,ni,ti,zeff,q,R,eps,varargin)
```

```
zion=ones(size(ne));
nargeff=nargin-8;
if nargeff > 0
   if ~isempty(varargin{1}); zion=varargin{1}; end
end

ii=find(ne>0 & te>0);
lnLam=zeros(size(ne));
lnLam(ii) = 31.3 - log(sqrt(ne(ii))./te(ii));

ii=find(ni>0 & ti>0);
lnLami=zeros(size(ni));
lnLami(ii) = 30. - log(zion(ii).^3.*sqrt(ni(ii))./ti(ii).^1.5);

ii=find(eps~=0);
nuestar=zeros(size(ne));
nuistar=zeros(size(ne));
nuestar(ii) = 6.921E-18 .* q(ii) .* R(ii) .* ne(ii) .* zeff(ii) .* lnLam(ii) ./ (te(ii).^2 .*
eps(ii).^1.5);
nuistar(ii) = 4.900E-18 .* q(ii) .* R(ii) .* ni(ii) .* zion(ii).^4 .* lnLami(ii) ./ (ti(ii).^2 .*
eps(ii).^1.5);
ii=find(eps==0);
nuestar(ii)=2.*zeff(ii);
nuistar(ii)=2.*zeff(ii);
end

function [signeo, sigspitzer] = sigmaneo(ft,ne,te,varargin)

nuestar=zeros(size(ft));
zeff=3. .* ones(size(ft));

nargeff=nargin-3;
if nargeff > 0
   if ~isempty(varargin{1}); zeff=varargin{1}; end
end
if nargeff > 1
   if ~isempty(varargin{2}); nuestar=varargin{2}; end
```

```
end

NZ = 0.58 + 0.74 ./ (0.76 + zeff);
lnLam = 17.*ones(size(ft));
ii=find(ne>0 & te>0);
if length(ii)>0
   lnLam = 31.3 - log(sqrt(ne)./te);
end

sigspitzer = 1.9012E+04 .* te.^1.5 ./ zeff ./ NZ ./ lnLam;

ft33eff = ft ./ (1. + (0.55-0.1.*ft).*sqrt(nuestar) + 0.45.*(1.-ft).*nuestar./zeff.^1.5);
signeo = sigspitzer .* (1. - ft33eff.*(1.+0.36./zeff - ft33eff.*(0.59./zeff -
0.23./zeff.*ft33eff)));
end


function [jB,L31,L32,L34,alfa] = jdotB_BS(ft,dln_ne,dln_te,dln_ti,peop,varargin)

zeff=3. .* ones(size(ft));
nuestar=zeros(size(ft));
nuistar=zeros(size(ft));
ptot=ones(size(ft));
Tpsi=ones(size(ft));

nargeff=nargin-5;
if nargeff > 0
   if ~isempty(varargin{1}); zeff=varargin{1}; end
end
if nargeff > 1
   if ~isempty(varargin{2}); nuestar=varargin{2}; end
end
if nargeff > 2
   if ~isempty(varargin{3}); nuistar=varargin{3}; end
end
if nargeff > 3
   if ~isempty(varargin{4}); ptot=varargin{4}; end
```

```
end
if nargeff > 4
   if ~isempty(varargin{5}); Tpsi=varargin{5}; end
end

[L31, L32, L34, alfa] = BScoeff(ft,zeff,nuestar,nuistar);

jB = - Tpsi.*ptot.* (L31.*dln_ne + peop.*(L31+L32).*dln_te + (1.-
peop).*(L31+alfa.*L34).*dln_ti);
end

function [L31, L32, L34, alfa] = BScoeff(ft,varargin)

% dim
dims=size(ft);
dimrho=dims(1);
dimt=dims(2);
if dimrho==1
   dimrho=dimt;
   dimt=1;
end
dims=[dimrho dimt];
ft=reshape(ft,dims);

zeff=3. * ones(dims);
nuestar=0.*ones(dims);
nuistar=0.*ones(dims);

nargeff=nargin-1;
if nargeff > 0
   if ~isempty(varargin{1}); zeff=reshape(varargin{1},dims); end
end
if nargeff > 1
   if ~isempty(varargin{2}); nuestar=reshape(varargin{2},dims); end
end
if nargeff > 2
   if ~isempty(varargin{3}); nuistar=reshape(varargin{3},dims); end
```

```
end

%    effective trapped fractions
sqnuestar=sqrt(nuestar);

ft31eff = ft ./ (1.+(1.-0.1.*ft).*sqnuestar + 0.5.*(1.-ft).*nuestar./zeff);
ft32ee_eff = ft ./ (1. + 0.26.*(1.-ft).*sqnuestar + 0.18.*(1.-0.37.*ft).*nuestar./sqrt(zeff));
ft32ei_eff = ft ./ (1. + (1.+0.6.*ft).*sqnuestar + 0.85.*(1.-0.37.*ft).*nuestar.*(1.+zeff));
ft34eff = ft ./ (1.+(1.-0.1.*ft).*sqnuestar + 0.5.*(1.-0.5.*ft).*nuestar./zeff);
alfa0 = - 1.17.*(1.-ft) ./ (1.-0.22.*ft-0.19.*ft.^2);

% coefficients
zeffp1 = zeff+1.;
L31 = ft31eff .* ( (1.+1.4./zeffp1) ...
      - ft31eff .* (1.9./zeffp1 - ft31eff .* (0.3./zeffp1 + 0.2./zeffp1 .* ft31eff)));
L32 = (0.05+0.62.*zeff)./zeff./(1.+0.44.*zeff).*(ft32ee_eff-ft32ee_eff.^4) ...
      + ft32ee_eff.^2.*(1.-1.2.*ft32ee_eff+0.2.*ft32ee_eff.^2) ./ (1.+0.22.*zeff) ...
      - (0.56+1.93.*zeff)./zeff./(1.+0.44.*zeff) .* (ft32ei_eff-ft32ei_eff.^4) ...
      + ft32ei_eff.^2.*(1.-0.55.*ft32ei_eff-0.45.*ft32ei_eff.^2) .* 4.95 ./ (1.+2.48.*zeff) ...
      + 1.2 ./ (1.+0.5.*zeff) .* (ft32ee_eff.^4-ft32ei_eff.^4);
L34 = ft34eff.* ( (1.+1.4./zeffp1) - ft34eff.*(1.9./zeffp1-
ft34eff.*(0.3./zeffp1+0.2./zeffp1.*ft34eff)) );
sqnui = sqrt(nuistar);
nui2ft6 = nuistar.^2 .* ft.^6;
alfa = ((alfa0 + 0.25.*(1.-ft.^2).*sqnui) ./ (1.+0.5.*sqnui) + 0.315.*nui2ft6) ./
(1.+0.15.*nui2ft6);
end

function ft = ftav_with_delta(eps, delta)

if isscalar(delta)
      delta = delta .* ones(size(eps));
end

% Step 1: Calculate effective inverse aspect ratio
epsilon_eff = 0.67 .* (1 - 1.4 * delta .* abs(delta)) .* eps;
```

```
% Step 2: Calculate ft using refined formula
numerator = (1 - epsilon_eff);
denominator = (1 + 2 .* sqrt(epsilon_eff));
sqrt_term = sqrt((1 - eps) ./ (1 + eps));

ft = 1 - (numerator ./ denominator) .* sqrt_term;

% Step 3: Ensure ft does not exceed 1
ft = min(ft, 1);
ft = max(ft, 0); % also clip to 0 if any negative due to numerical error
end
```

A.15 Formosa Integrated Research Spherical Tokamak (FIRST)

In the study, we developed a series of numerical models for calculating key physical quantities in the mini-Tokamak, including the resistance and inductance of the components, the eddy currents in the vacuum-vessel walls, the required and induced loop voltages, as well as plasma parameters such as temperature, current, ionization fraction, and resistance. These models were primarily designed to support the optimization of the central solenoid current profile. Here, we would like to apply the same models to Formosa Integrated Research Spherical Tokamak (FIRST), the first tokamak that is being built in Taiwan.

This appendix consists of three parts. Appendix 15.1 introduces the specifications of FIRST, including the vacuum-vessel geometry, the central solenoid configuration, and the desired plasma shape. Appendix 15.2 presents the calculation of the required breakdown voltage, and Appendix 15.3 shows the calculation of plasma parameters. Finally, Appendix 15.4 provides the conclusion.

A.15.1 Specification

Formosa Integrated Research Spherical Tokamak (FIRST) is the first tokamak being developed in Taiwan. The expected date of first tokamak plasma is in 2026. The major and minor radii of plasma in FIRST are 450 mm and 320 mm, respectively. The elongation will be 2.4 so that the long and short axis of the plasma cross section is 768 mm and 320 mm, respectively. The vacuum vessel is approximately elliptical in shape, featuring an arc radius of 968 mm as shown in Figure 65(a). The center of the arc is 140 mm away from the z axis of the system as shown in Figure 65(b). The device will provide a toroidal magnetic field of up to 0.5 T at R = 450 mm, the goal of the plasma current is 100 kA. Achieving a temperature of 100 eV is the first milestone of the project.

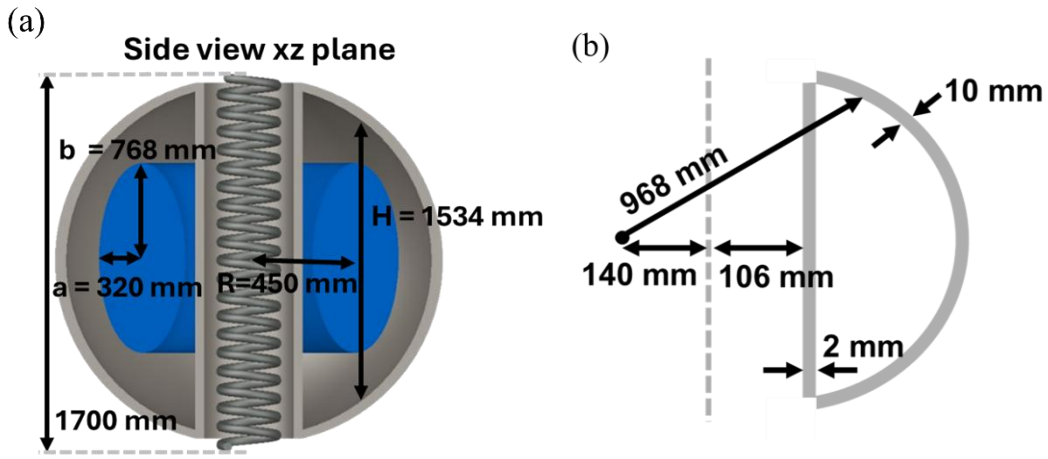

Figure 65: (a) Simpled xz-plane cross-section of FIRST. (b) Specifications of the vacuum vessel.

Figure 66(a) shows the central solenoid in FIRST, which consists of two layers with 140 turns each, resulting in a total of 280 turns. The distances from the centerline to the centers of the inner and outer layers are 88 mm and 100 mm, respectively. For simplicity, the major radius of the central solenoid in our calculations is set as their average, 94 mm. The total height of the solenoid is 1680 mm.

The solenoid coil is a hollow copper tube wrapped with an insulating layer on the outside. The blue circle represents the insulation layer, which is 1 mm thick. The black circle represents the coil, with a diameter of 10 mm. The gray circle represents the hollow section for cooling water, with a diameter of 7.6 mm. Including the insulation, the total diameter of a single turn is therefore 12 mm.

For the vacuum vessel, we divide it into two parts: the inner-vacuum-vessel wall and the outer-vacuum-vessel wall, as shown in Figure 65(b). The curve of the outer-vacuum-vessel wall, which is approximately elliptical in shape, centered at x = -140 mm, with an arc radius of 968 mm, is calculated based on the equation:

$$(r + 140)^2 + z^2 = 968^2. \tag{129}$$

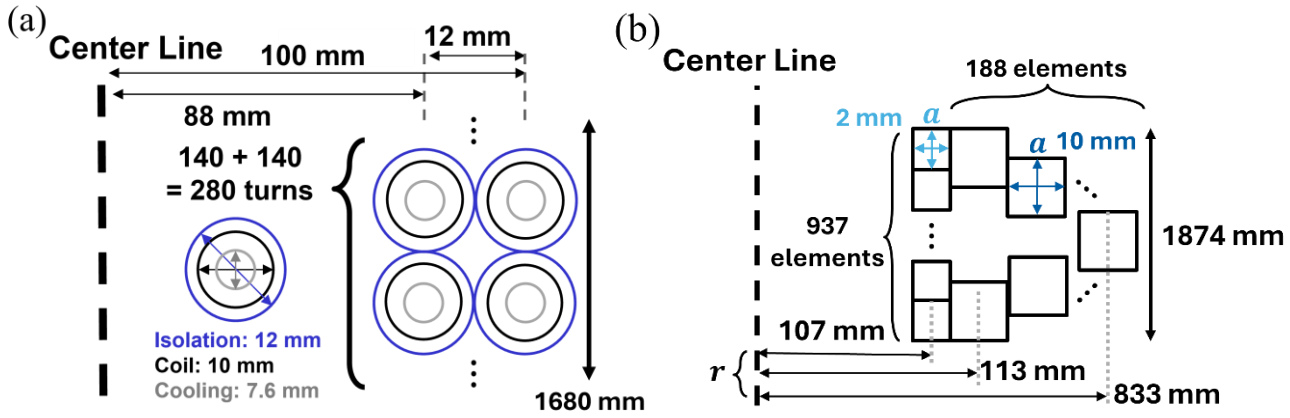

Figure 66: (a) Specifications of the central solenoid. (b) Specifications of the inner and outer vacuum-vessel walls divided for calculation.

The vacuum-vessel wall is treated as many ring coils stacking on top of each other. To enhance calculation accuracy, as illustrated in Figure 66(b), we divide the inner-vacuum-vessel wall into 937 identical ring coils with square cross-sections of 2 mm in each side, stacked together. Similarly, the outer-vacuum-vessel wall is divided into 188 ring coils with square cross-sections of 10 mm in each side and with different ring radius, stacked together. The radius of each element is calculated using Eq. (129) This segmentation allows for precise modeling of the eddy currents generated within the vacuum-vessel walls. All components are implemented in MATLAB according to the specifications, and the detailed code is provided in Appendix A.16.

A.15.2 Required loop voltage for breakdown

To design an appropriate solenoid current profile, it is first necessary to determine the breakdown conditions. With the specifications of the central solenoid and vacuum-vessel walls, we apply the models developed in the previous chapters to calculate the key physical quantities in FIRST. Finally, we design the central solenoid current profile for generating the required loop voltage including considering the eddy current induced in the vacuum-vessel wall.

First, we calculate the required loop voltage for breakdown. To do so, we must determine the connection length in FIRST. This calculation follows the same method described in Section 4.1. Since the shape of the outer-vacuum-vessel wall is already known as an arc centered at x = -140 mm with an arc radius of 968 mm, as shown in Figure 67:

$$(\mathrm{x}+140)^2+\mathrm{z}^2=968^2. \tag{130}$$

The vertical side equals the vacuum vessel height at the location 450 mm from the centerline. It can be calculated as

$$y=\sqrt{968^2-(450+140)^2}=767\ (\text{at x}=450\text{mm}),$$
$$\Rightarrow 2y=1534\ \text{mm}. \tag{131}$$

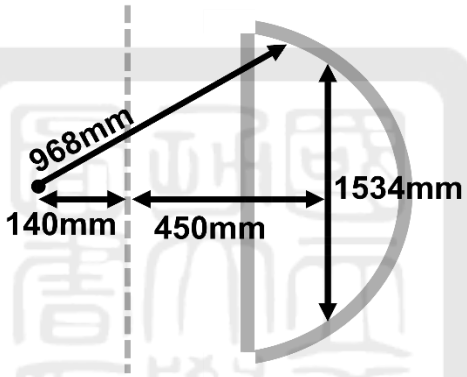

Figure 67: Specifications of the vacuum vessel.

This vertical distance of 1534 mm corresponds to the total extent of the electron's motion in the z direction, as shown in Figure 67. Assuming a magnetic field ratio B_Z/B_T = 1/1000, the corresponding connection length is estimated to be 1534 m.

Using the same method described in Section 4.1, the required loop voltage for breakdown is calculated based on the empirical Townsend criterion:

$$E_{\mathrm{BD}}=\frac{BP}{\ln(APL)} \tag{132}$$

where P is the prefill gas pressure, L is the connection length, and A, B are gas-specific constants. Unlike the mini-Tokamak, which uses helium, FIRST uses hydrogen as the

prefill gas. Therefore, the constants are taken as $A = 3.83\ \mathrm{m^{-1}\cdot Pa^{-1}}$ and $B = 93.6\ \mathrm{V\cdot m^{-1}\cdot Pa^{-1}}$.

Then, we calculate the breakdown electric field under specified conditions, which depend on parameters such as gas pressure, connection length, and the magnetic field ratio B_T/B_Z. Figure 68(a) shows the breakdown curve of the electric field for a gas pressure range from 10^{-4} Pa to 10^{0} Pa, with the ratio between B_T and B_Z varying from 0.1 to 0.001.

Multiplying the electric field by the total field line length, we can further determine the breakdown voltage (V_{BD}) at our desired plasma position (R = 450 mm):

$$V_{BD} = E_{BD} \cdot 2\pi R, \tag{133}$$

as shown in Figure 68(b).

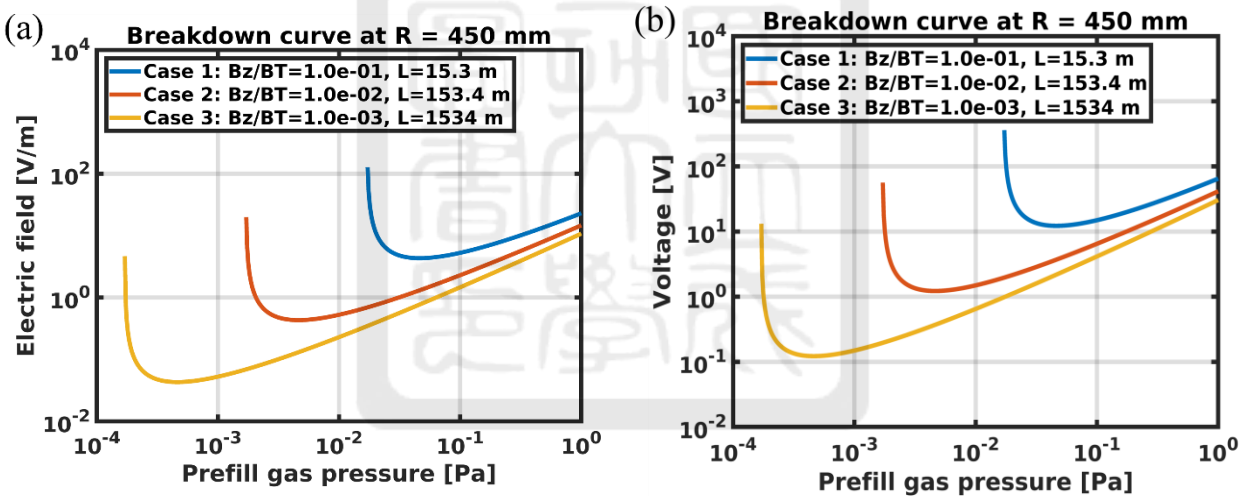

Figure 68: (a) Breakdown electric field curve. (b) Breakdown voltage curve.

Using the previously described method in Section 4.2, we calculate the breakdown voltage under the target conditions of our experiment. For example, when B_Z is 0.1% of B_T, and the vacuum vessel height is 1534 mm (at a radius of 450 mm), the connection length (L) is 1534 m. Additionally, if the pressure is 10^{-4} Torr (equivalent to 1.3×10^{-2} Pa), i.e., the

particle density is approximately 10^{18} m^{-3}, the breakdown electric field is calculated using Eq. (132):

$$E = \frac{BP}{\ln(APL)} = \frac{93.6 \times 1.3 \times 10^{-2}}{\ln(3.83 \times 1.3 \times 10^{-2} \times 1534)} = 0.28\ {}^{\mathrm{V}}/_{\mathrm{m}}. \quad (134)$$

Based on this electric field, we can calculate the breakdown voltage required to generate plasma at a distance of 0.45 m from the centerline using Eq. (133):

$$V = E \cdot 2\pi R = 0.28 \times 2\pi \times 0.45 = 0.8\ \mathrm{V}. \quad (135)$$

To induce this loop voltage in the vacuum vessel, we calculate the required rate of change of the central solenoid current using Faraday's law. Substituting the solenoid geometry and parameters, length $l = 1.68$ m, radius $r = 0.094$ m, cross-sectional area $A = \pi r^2 = \pi(0.094)^2 = 0.314\ \mathrm{m}^2$, number of turns $N = 280$, and permeability of free space $\mu_0 = 4\pi \times 10^{-7}$ H/m, into Eq. (70):

$$\frac{dI}{dt} = -\frac{0.8 \times 1.68}{\pi(0.094)^2 \cdot 280 \cdot 4\pi \times 10^{-7}} = -138\ {}^{\mathrm{A}}/_{\mathrm{ms}}. \quad (136)$$

Once the current rate is determined, a current profile over a specified duration, e.g., 15 ms, can be generated. As shown by the blue solid line in Figure 69(a), the current drops from 2.07 kA to 0 kA in 15 ms. This profile is then used in conjunction with the model developed in Chapter 3 to calculate the eddy currents induced in the inner and outer vacuum-vessel walls by the time-varying central solenoid current. As illustrated in Figure 69(a), the total eddy currents in the inner and outer walls are represented by the red solid and red dashed lines, respectively. The total eddy currents in the inner and outer walls reach ~5 kA and ~4 kA, respectively. These total currents are obtained by summing the contributions from each individual wall segment, as shown in Figure 69(b) and (c).

The induced loop voltage is often several times higher than the calculated breakdown voltage since the connecting length we used was an optimistic condition where $B_Z/B_T = 0.1$ %. Using this program, we can quickly calculate the maximum central solenoid current

required for different loop voltages and durations. As shown in Table 7, if we aim to induce a loop voltage of 10 V, the maximum current of the central solenoid would need to be 20 kA, which is very challenging. Alternatively, we can reduce the time duration (t_{duratin}) while maintaining the same current rate ($I_{\text{max}}/t_{\text{duratin}}$) to lower the required maximum current of the central solenoid. For example, if t_{duratin} is 10 ms, the $I_{\text{max}} = 17$ kA. It's a more reasonable number.

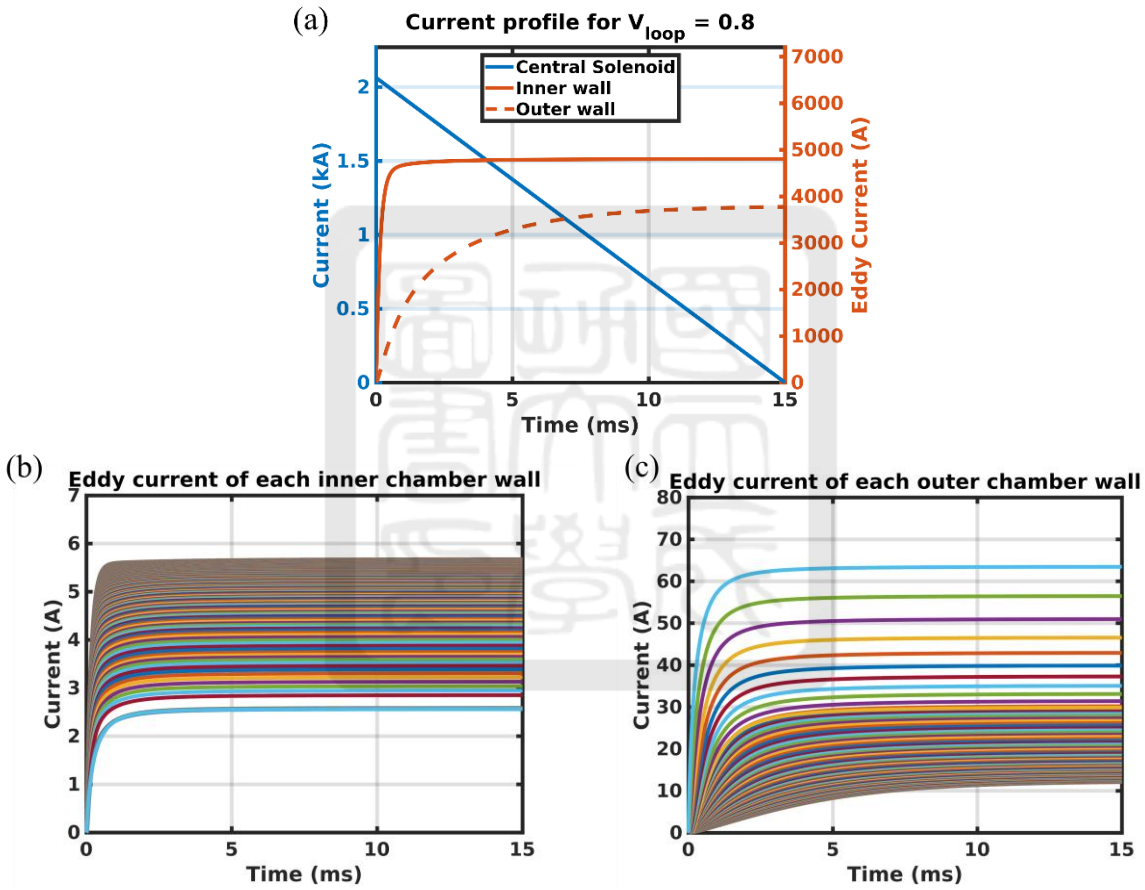

Figure 69: (a) Current profile of central solenoid and the eddy currents of the inner and outer vacuum-vessel wall. (b) The eddy current of each element of the inner-vacuum-vessel wall. (c) The eddy current of each element of the outer-vacuum-vessel wall.

Table 7: Central solenoid current requirements for different loop voltages and durations.

V_{loop}	$t_{duratin}$	I_{max}
0.8 V	15 ms	2.1 kA
5 V	15 ms	13 kA
10 V	10 ms	17 kA
10 V	15 ms	26 kA

With this current profile, we then computed the magnetic field distribution produced by each coil and integrated the resulting fields to obtain the total magnetic flux in the equatorial plane. Based on the flux variation over time, the loop voltage was derived using Eq. (64), as shown in Figure 70.

It can be observed that around 6.5 ms, the influence of eddy currents in the vacuum-vessel walls on the loop voltage reduces to approximately 5%. It indicates that the impact of eddy currents in the vacuum-vessel walls on the loop voltage can be ignored for a central solenoid current profile with a duration longer than 6.5 ms.

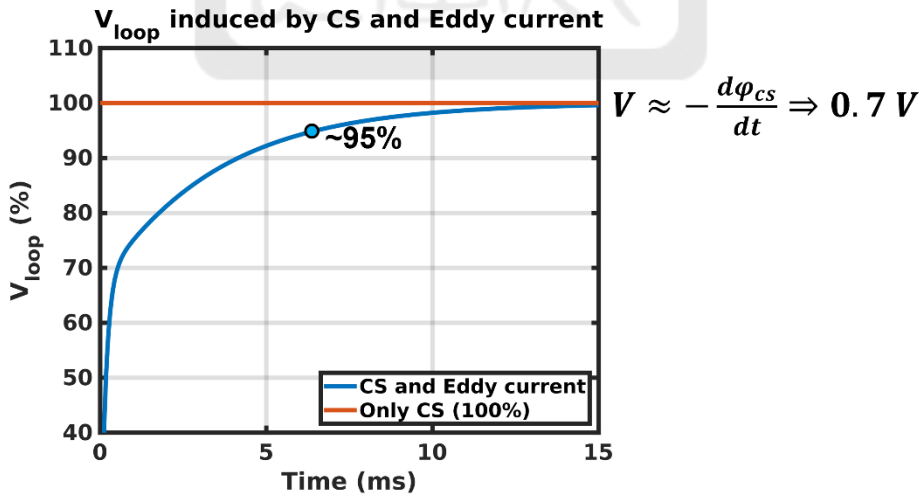

Figure 70: Comparison of loop voltage induced by central solenoid and eddy currents in the vacuum-vessel walls over time.

A.15.3 Plasma parameters

Lastly, we calculate the plasma parameters by providing the initial conditions, including the initial plasma temperature, working gas pressure, and the central solenoid current profile. Using these inputs, the plasma parameters at each time step are computed based on the model in Chapter 5.

Three cases were considered: (a) Gas density of 10^{17} m^{-3}, with initial temperature of 0.026 eV; (b) Gas density of 10^{17} m^{-3}, with initial temperature of 1 eV; (a) Gas density of 10^{18} m^{-3}, with initial temperature of 5 eV. They are listed in Table 8.

Table 8: Plasma startup scenarios and central solenoid current profiles in FIRST.

Case	n_0	T_0	$I_{\mathrm{cs\_max,1}}$	$\Delta t_{\mathrm{cs\_1}}$	$I_{\mathrm{cs\_max,2}}$	$\Delta t_{\mathrm{cs\_2}}$	$V_{\mathrm{loop\_1}}$	$V_{\mathrm{loop\_2}}$
a	10^{17} m^{-3}	0.026 eV	1.7 kA	10 ms	-1.08 kA	40 ms	1 V	0.16 V
b	10^{18} m^{-3}	0.026 eV	10 kA	5 ms	-10 kA	45 ms	11.6 V	1.3 V
c	10^{18} m^{-3}	5 eV	10 kA	5 ms	-10 kA	45 ms	11.6 V	1.3 V

(a) The initial plasma temperature is set to 0.026 eV, and the gas density is assumed to be 10^{17} m^{-3}, corresponding to a pressure of approximately 10^{-5} Torr, the results are shown in Figure 71(a). The applied central solenoid current profile, represented by the blue solid line in Figure 71(a), decreases linearly from 1.7 kA to 0 kA over the first 10 ms, followed by a linear ramp from 0 kA to −1.08 kA over the next 40 ms. This time-varying current induces a loop voltage of approximately 1 V over the first 10 ms, which exceeds the required breakdown voltage of 0.06 V at R = 450 mm under a gas pressure of 10^{-5} Torr.

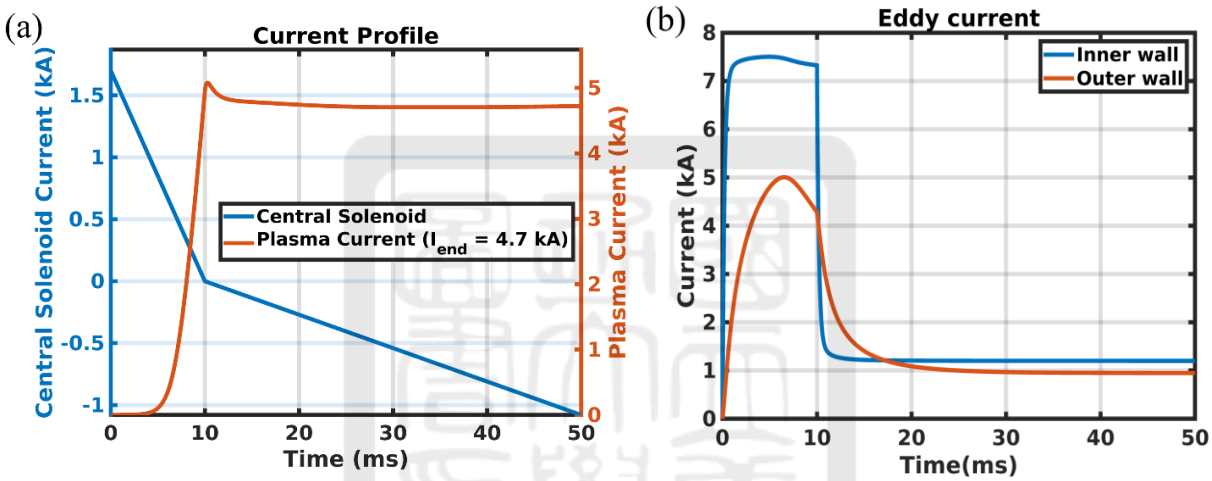

Figure 71: (a) Central solenoid and plasma current. (b) Current profile of inner and outer vacuum-vessel walls.

Figure 71(a) shows the plasma current in the red line, induced by the central solenoid current profile in the blue line, while Figure 71(b) presents the total eddy currents in the inner and outer vacuum-vessel walls, respectively. The target plasma parameters for FIRST are a plasma temperature of 100 eV and a plasma current of 100 kA. However, compared to achieving high plasma current, our current focus is to reach and sustain the target plasma temperature at 100 eV, as shown in Figure 72(a). Figure 72(b) shows the time evolution of the ionization fraction. The plasma becomes fully ionized at approximately 11 ms and remains at this level.

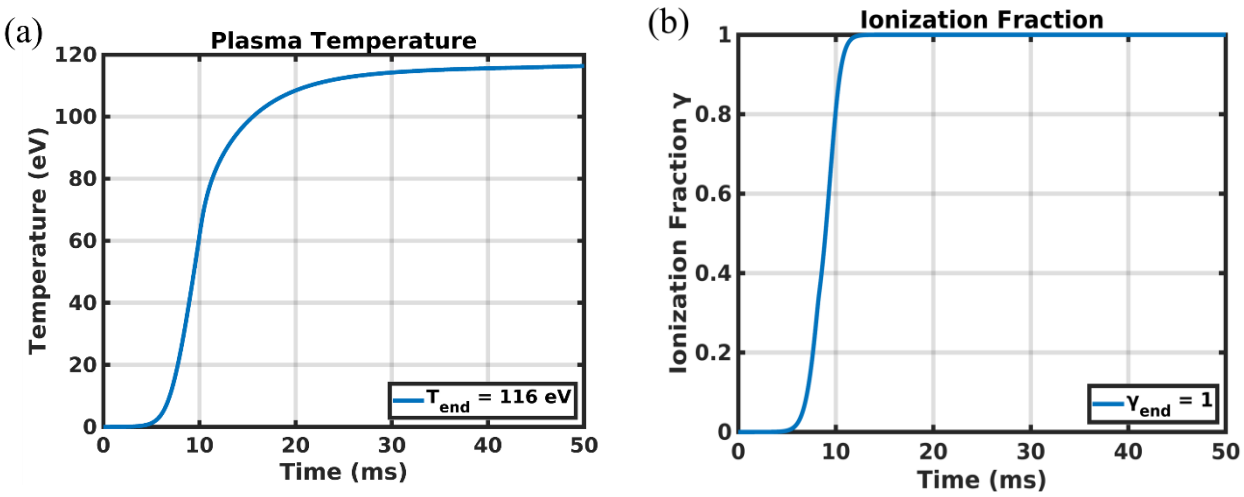

Figure 72: Plasma temperature. (b) Ionization fraction.

(b) The initial plasma temperature is set to 0.026 eV, and the gas density of 10^{18} m^{-3}, corresponding to a pressure of approximately 10^{-4} Torr. The applied central solenoid current profile, represented by the blue solid line in Figure 73(a), decreases linearly from 10 kA to 0 kA over the first 5 ms, followed by a linear ramp from 0 kA to −10 kA over the next 45 ms. This time-varying current induces a loop voltage of approximately 11.6 V over the first 5 ms, which exceeds the required breakdown voltage of 0.8 V at R = 450 mm under the same pressure condition.

As shown in Figure 73(a), the plasma current, represented by the red solid line, driven by the sharp initial change rate of the central solenoid current reaches its peak value of approximately 78 kA at around 5.3 ms. However, as the rate of change of the solenoid current decreases after 5 ms, the induced loop voltage also declines, leading to a gradual reduction in plasma current. Figure 73(b) presents the total eddy currents in the inner and outer vacuum-vessel walls induced by the central solenoid current profile.

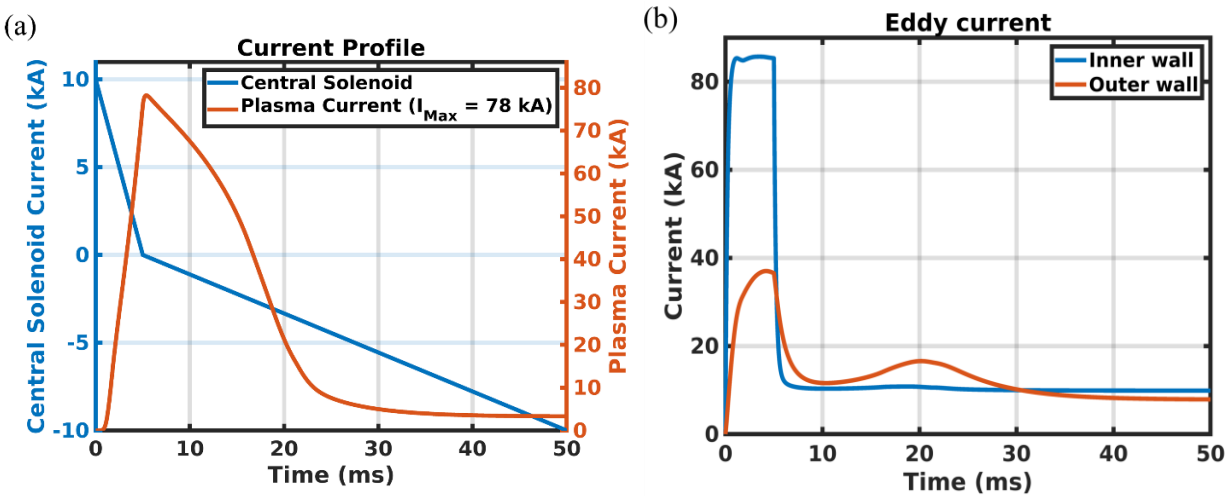

Figure 73: (a) Central solenoid and plasma current. (b) Current profile of inner and outer vacuum-vessel walls.

Figure 74(a) shows the plasma temperature evolution. The red dashed vertical lines indicate the period during which the plasma remains fully ionized, which lasts for approximately 10 ms. The plasma temperature reaches a peak value of approximately 125 eV at around 5.8 ms, due to the sharp initial change rate of the central solenoid current. However, as the rate of change of the solenoid current decreases after 5 ms, the induced plasma current declines, resulting in insufficient Ohmic heating and a gradual reduction in plasma temperature.

Figure 74(b) presents the time evolution of the plasma ionization fraction. The plasma becomes fully ionized at around 3.65 ms, and maintains this state until approximately 16 ms. Afterward, the ionization fraction begins to decrease, following the same general trend as the plasma temperature.

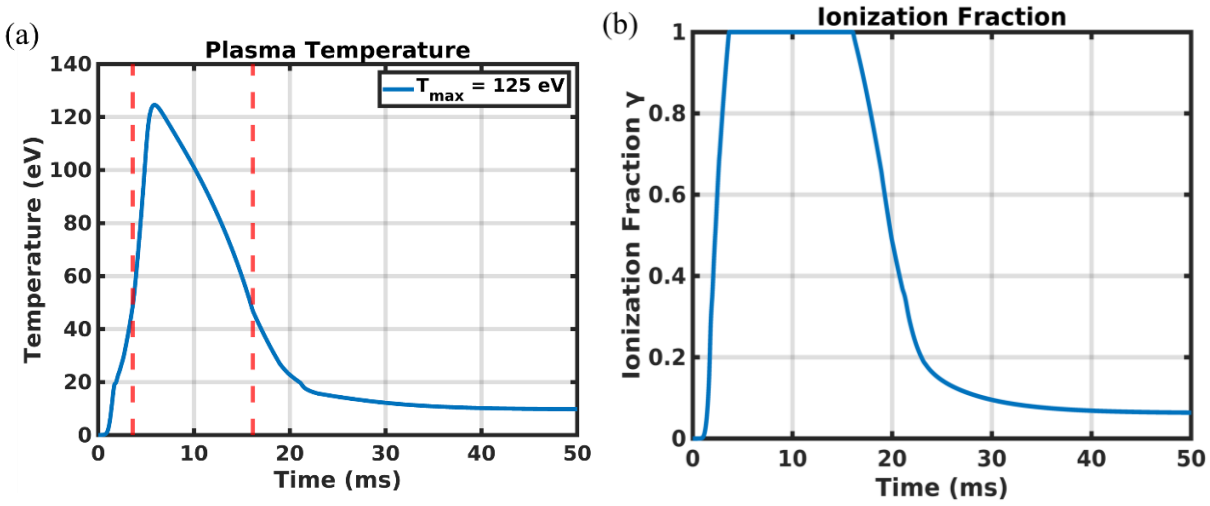

Figure 74: Plasma temperature. (b) Ionization fraction.

(c) The initial plasma temperature is set to 5 eV, and the gas density is assumed to be 10^{18} m^{-3}. The central solenoid current profile decreases linearly from 10 kA to 0 kA over the first 5 ms, followed by a linear ramp from 0 kA to −10 kA over the next 45 ms. It is the same as that in case (b) and is shown in Figure 75(a). Preheating enables the plasma current, represented by the red solid line, to reach a higher peak value of approximately 93 kA at around 5.3 ms.

Figure 75(b) presents the total eddy currents in the inner and outer vacuum-vessel walls for case (b) and (c). There is no significant difference in the eddy current behavior, as it is primarily determined by the rate of change of the central solenoid current, which remains the same in both cases. However, a noticeable difference can be observed in the eddy current of the inner-vacuum-vessel wall. Due to the delayed decay of the plasma current in this case as shown in Figure 75(a), the secondary rise of the eddy current in the inner-vacuum-vessel wall, caused by the changing magnetic field from the decaying plasma current, occurs later. Specifically, the peak of this secondary feature shifts from 20 ms to approximately 31 ms. A similar trend can also be observed in the eddy current of outer-vacuum-vessel wall, although the effect is less pronounced due to its larger distance from the plasma.

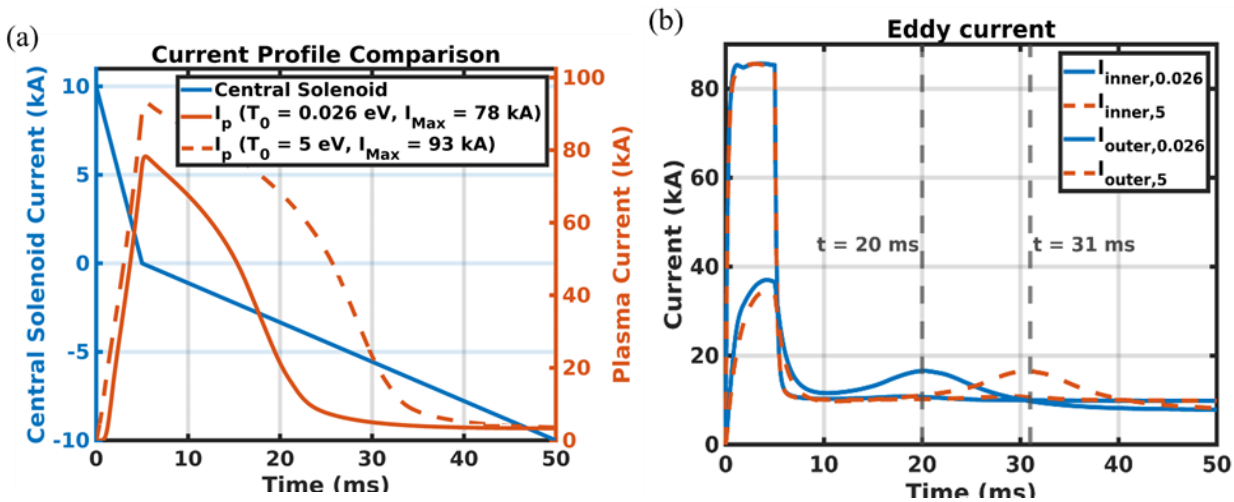

Figure 75: (a) Central solenoid and plasma current. (b) Current profile of inner and outer vacuum-vessel walls.

Figure 76(a) and Figure 76(b) show the plasma temperature profile and plasma ionization fraction, respectively. The results demonstrate that preheating not only increases the peak plasma temperature, reaching approximately 165 eV at around 6.2 ms due to the enhanced plasma current and resulting stronger Ohmic heating, but also extends the fully ionized duration. The temperature increases ~32% compare to the no-preheated case in case (b). The plasma becomes fully ionized at approximately 3 ms and maintains this state until around 26.4 ms, representing an extension of about 11 ms compared to the non-preheated case, i.e., 89% increase. These results highlight the effectiveness and importance of preheating in achieving and sustaining plasma conditions. Therefore, the initial condition in case (c) is the most preferable condition.

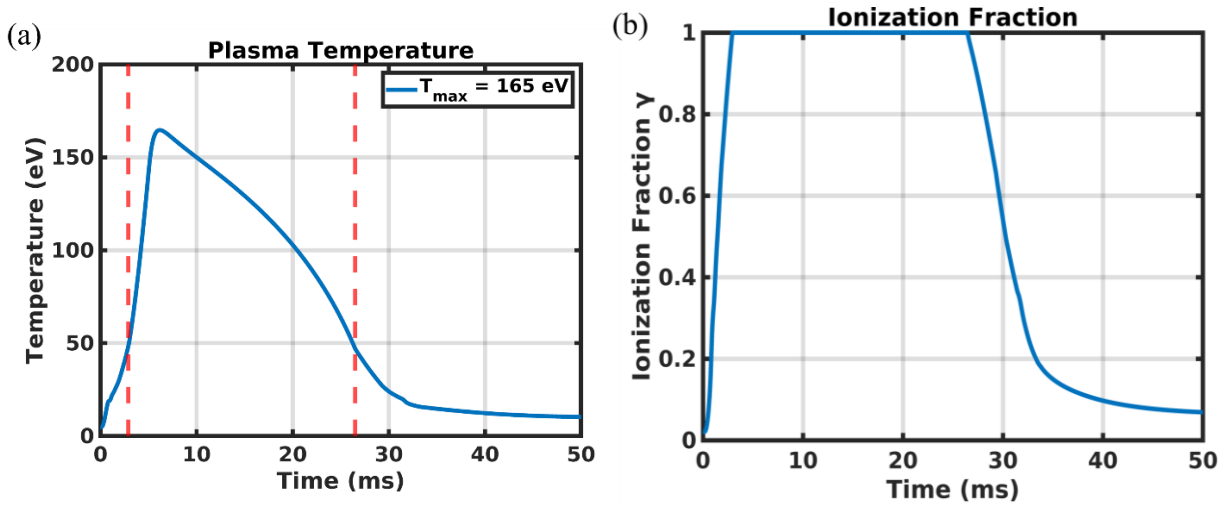

Figure 76: Plasma temperature. (b) Ionization fraction.

A.15.4 Conclusion

In this section, we use our model to design the central solenoid current profile for the Formosa Integrated Research Spherical Tokamak (FIRST) and propose three different cases. One case targets an initial plasma density of n = 10^{17} m^{-3}, aiming to maintain a plasma temperature of approximately 100 eV for 35 ms. The other two cases are designed for a higher density of n = 10^{18} m^{-3}, both aiming to reach 100 eV. We find that introducing a 5 eV preheat increases the peak plasma temperature from 125 eV to 165 eV, an improvement of approximately 32%, and extends the fully ionized duration from 12.35 ms to 23.4 ms, representing an 89% increase. Based on these results, case (c) in Table 9, which includes preheating, is preferred, as it significantly increases both the peak temperature and the duration of full ionization. These enhancements contribute to improved plasma startup quality and stability.

Table 9: Plasma startup scenarios and central solenoid current profiles in FIRST.

Case	n_0	T_0	$I_{cs\_max,1}$	Δt_{cs_1}	$I_{cs\_max,2}$	Δt_{cs_2}	V_{loop_1}	V_{loop_2}
a	10^{17} m^{-3}	0.026 eV	1.7 kA	10 ms	-1.08 kA	40 ms	1 V	0.16 V
b	10^{18} m^{-3}	0.026 eV	10 kA	5 ms	-10 kA	45 ms	11.6 V	1.3 V
c	10^{18} m^{-3}	5 eV	10 kA	5 ms	-10 kA	45 ms	11.6 V	1.3 V

A.16 Formosa Integrated Research Spherical Tokamak (FIRST)

```
% The code is in Student_Thesis/2025_ypi/Code/parameter_FIRST_thesis on the NAS.
% ========== Arc Mode ==========

%% ---- Enable Components ----
enable_chamber = true;
enable_solenoid = true;
enable_plasma = true;
enable_pfc = false;

%% ========== Input Parameters ==========

% Chamber
if enable_chamber
    % Arc chamber parameters
    offset_x = 140e-3;
    thickness = 10e-3;
    inner_radius = 968e-3 + thickness/2;
    radius = thickness/2;
    inner_thickness = 2e-3;
    inner_distance = 100e-3 + inner_thickness/2;
    % Arc geometry
    x_target = inner_distance + thickness / 2;
    cos_theta = (x_target + offset_x) / inner_radius;
    theta_start = -acos(cos_theta);
    theta_end = acos(cos_theta);
    z_top = inner_radius * sin(theta_end);
    z_bottom = inner_radius * sin(theta_start);
    chamber_height = abs(z_top - z_bottom);

    % Inner wall params
    inner_params.height = chamber_height;
    inner_params.thickness = inner_thickness;
    inner_params.distance = inner_distance;
    % Outer wall params
    outer_params = struct( ...
```

```
            "height", chamber_height, ...
            "radius", radius, ...
            "thickness", thickness, ...
            "inner_radius", inner_radius, ...
            "offset_x", offset_x, ...
            "theta_start", theta_start, ...
            "theta_end", theta_end);

        % Generate wall geometry
        [inner_x, inner_z, chamber_components.inner_wall] =
generate_inner_wall(inner_params);
        [outer_x, outer_z, chamber_components.outer_wall] =
generate_outer_wall_arc(outer_params);

        % Chamber export table (same format as Rec Mode)
        num_inner = length(inner_x);
        num_outer = length(outer_x);
        names = [arrayfun(@(i) sprintf('Inner Wall %d', i), 1:num_inner, 'UniformOutput',
false)'; ...
                    arrayfun(@(i) sprintf('Outer Wall %d', i), 1:num_outer, 'UniformOutput',
false)'];
        r_values = [inner_x; outer_x];
        z_values = [inner_z; outer_z];
        thickness_values = [repmat(inner_params.thickness, num_inner, 1);
repmat(outer_params.thickness, num_outer, 1)];
        radius_values = [repmat(inner_params.thickness/2, num_inner, 1);
repmat(outer_params.radius, num_outer, 1)];
        coil_lengths = 2 * pi * r_values;

        chamber_table = table(names, r_values, z_values, thickness_values, radius_values,
coil_lengths, ...
            'VariableNames', {'Component', 'Distance_m', 'Z_Position_m', 'Thickness_m',
'Radius_m', 'Coil_Length_m'});
        save_table_with_overwrite(chamber_table, 'chamber_components.xlsx');
end

% Solenoid
```

```
if enable_solenoid
    solenoid_radius = 5e-3;
    solenoid_radius_cooling = 3.8e-3;
    isolation_thickness = 1e-3;
    solenoid_total_radius = solenoid_radius + isolation_thickness;
    solenoid_inner_num_coils = 140;
    solenoid_layer_count = 2;
    solenoid_outer_num_coils = solenoid_inner_num_coils * (solenoid_layer_count ==
2);
    solenoid_inner_distance = 82e-3;
    solenoid_outer_distance = solenoid_inner_distance + (solenoid_layer_count == 2) * 2
* solenoid_total_radius;
    solenoid_height_between_turns = 12e-3;
    solenoid_height = solenoid_total_radius * 2 * solenoid_inner_num_coils;
    solenoid_coil_A = pi * (solenoid_radius^2 - solenoid_radius_cooling^2);
    solenoid_a = pi * (solenoid_inner_distance - solenoid_radius)^2;

    solenoid_table = table({"solenoid"}, solenoid_radius, solenoid_radius_cooling,
isolation_thickness, solenoid_total_radius, ...
        solenoid_inner_num_coils, solenoid_outer_num_coils, solenoid_inner_distance,
solenoid_outer_distance, ...
        solenoid_height_between_turns, solenoid_height, solenoid_coil_A,
solenoid_a, ...
        'VariableNames', {'Component', 'Radius_m', 'Cooling_m', 'Isolation_Thick_m',
'Total_Radius_m', 'Inner_Num_Coils', 'Outer_Num_Coils', 'Inner_Distance_m',
'Outer_Distance_m', 'Height_Between_Turns_m', 'Total_Height_m', 'Coil_Area_m2',
'A_Param'});
    writetable(solenoid_table, 'solenoid_parameters.xlsx');
end

% Plasma
if enable_plasma
    T_plasma = 1; % eV
    n_total_plasma = 1e17; % m^-3
    a_plasma = 0.32;
    kappa = 2.4;
    b_plasma = a_plasma * kappa;
```

```
    R_plasma = 0.45;
    A_plasma = pi * a_plasma * b_plasma;
    L_plasma = 2 * pi * R_plasma;
    V_plasma = 2 * pi^2 * R_plasma * a_plasma * b_plasma;

    plasma_table = table({"plasma"}, T_plasma, n_total_plasma, a_plasma, kappa,
R_plasma, A_plasma, L_plasma, V_plasma, ...
        'VariableNames', {'Component', 'Temperature_eV', 'Particle_Density_m3',
'Semi_Minor_Axis_m', 'Kappa', 'Major_Radius_m', 'Cross_Sectional_Area_m2',
'Length_m', 'Volume_m3'});
    writetable(plasma_table, 'plasma_parameters.xlsx');
end

% PFC
coil_component = struct();
if enable_pfc
    pfc_list = {
        struct('name', 'PFC1', 'width', 11e-3, 'radius', inner_thickness/2, ...
                'distance', 850e-3 + inner_thickness/2, 'z_values', 0, 'num_coils', 1)
    };
    coil_component.PFC = struct();
    pfc_counter = 1;
    for k = 1:length(pfc_list)
        pfc = pfc_list{k};
        [pfc_x, ~, pfc_z] = setup_coil(pfc.num_coils, pfc.radius, pfc.width, ...
            pfc.distance, pfc.z_values, pfc.z_values, 0, 0);
        for i = 1:pfc.num_coils
            coil_component.PFC(pfc_counter).r = pfc_x(i);
            coil_component.PFC(pfc_counter).z = pfc_z(i);
            coil_component.PFC(pfc_counter).thickness = pfc.width;
            coil_component.PFC(pfc_counter).radius = pfc.radius;
            coil_component.PFC(pfc_counter).coil_length = 2 * pi * pfc.distance;
            pfc_counter = pfc_counter + 1;
        end
    end
    % Export table (same as Rec Mode)
    coil_names = fieldnames(coil_component);
```

```
    coil_table_data = [];
    for i = 1:length(coil_names)
        comp_name = coil_names{i};
        coils = coil_component.(comp_name);
        for j = 1:length(coils)
            coil_table_data = [coil_table_data; {
                sprintf('%s %d', comp_name, j), ...
                coils(j).r, coils(j).z, coils(j).thickness, coils(j).radius,
coils(j).coil_length
            }];
        end
    end
    coil_table = cell2table(coil_table_data, ...
        'VariableNames', {'Component', 'Distance_m', 'Z_Position_m', 'Thickness_m',
'Radius_m', 'Coil_Length_m'});
    writetable(coil_table, 'coil_components.xlsx');
end

%% ========== Plotting ==========

figure;
hold on;

if enable_chamber
    plot(outer_x, outer_z, 'ro', 'MarkerFaceColor', 'r', 'DisplayName', 'Outer Wall');
    plot(inner_x, inner_z, 'bo', 'MarkerFaceColor', 'b', 'DisplayName', 'Inner Wall');
end

if enable_plasma
    theta = linspace(0, 2*pi, 200);
    x_plasma = R_plasma + a_plasma * cos(theta);
    z_plasma = b_plasma * sin(theta);
    plot(x_plasma, z_plasma, 'k-', 'DisplayName', 'Plasma');
end

if enable_solenoid
    solenoid_z = linspace(-solenoid_height/2 + solenoid_total_radius, ...
```

```
                                    solenoid_height/2 - solenoid_total_radius,
solenoid_inner_num_coils)';
    plot(repmat(solenoid_inner_distance, solenoid_inner_num_coils, 1), solenoid_z, ...
        'gx', 'DisplayName', 'Solenoid Inner');
    if solenoid_layer_count == 2
        plot(repmat(solenoid_outer_distance, solenoid_outer_num_coils, 1),
solenoid_z, ...
            'mx', 'DisplayName', 'Solenoid Outer');
    end
end

if enable_pfc
    for p = 1:length(coil_component.PFC)
        plot(coil_component.PFC(p).r, coil_component.PFC(p).z, ...
            'go', 'MarkerSize', 6, 'MarkerFaceColor', 'g', 'DisplayName', 'PFC');
    end
end

xlabel('X Position (m)');
ylabel('Z Position (m)');
title('Chamber XZ Plane View');
legend('Location', 'bestoutside');
set(gca, 'linewidth', 1.1, 'fontsize', 14, 'LineWidth', 1.5, 'FontWeight', 'bold');
grid on;
hold off;
saveas(gcf, 'wall_arc_plot.png');

%% ====== Function Block ======
function [x_values, z_values, wall_struct] = generate_inner_wall(params)
    radius = params.thickness / 2;
    num_coils = ceil(params.height / params.thickness);
    z_values = linspace(-params.height/2 + radius, params.height/2 - radius, num_coils)';
    x_values = repmat(params.distance, num_coils, 1);
    wall_struct = repmat(struct('r',0,'z',0,'thickness',0,'radius',0,'coil_length',0), num_coils,
1);
    for i = 1:num_coils
        wall_struct(i).r = x_values(i);
```

```
            wall_struct(i).z = z_values(i);
            wall_struct(i).thickness = params.thickness;
            wall_struct(i).radius = radius;
            wall_struct(i).coil_length = 2 * pi * x_values(i);
        end
    end

    function [x_values, z_values, wall_struct] = generate_outer_wall_arc(params)
        radius = params.radius;
        num_coils = ceil(params.height / params.thickness);
        theta_values = linspace(params.theta_start, params.theta_end, num_coils)';
        x_values = params.inner_radius * cos(theta_values) - params.offset_x;
        z_values = params.inner_radius * sin(theta_values);
        wall_struct = repmat(struct('r',0,'z',0,'thickness',0,'radius',0,'coil_length',0), num_coils,
    1);
        for i = 1:num_coils
            wall_struct(i).r = x_values(i);
            wall_struct(i).z = z_values(i);
            wall_struct(i).thickness = params.thickness;
            wall_struct(i).radius = radius;
            wall_struct(i).coil_length = 2 * pi * x_values(i);
        end
    end

    function [x_values, y_values, z_values] = setup_coil(num_coils, coil_radius,
    coil_thickness, ...
        coil_distance, z_start, z_end, theta_start, theta_end, offset_x, use_sin_z)
        if nargin < 9, offset_x = 0; end
        if nargin < 10, use_sin_z = false; end
        theta_values = linspace(theta_start, theta_end, num_coils)';
        x_values = coil_distance * cos(theta_values) + offset_x;
        y_values = coil_distance * sin(theta_values);
        if use_sin_z
            z_values = coil_distance * sin(theta_values);
        else
            z_values = linspace(z_start, z_end, num_coils)';
        end
```

```
end

function save_table_with_overwrite(table_data, file_name)
    if isfile(file_name)
        delete(file_name);
    end
    writetable(table_data, file_name);
    fprintf('File "%s" has been saved successfully (overwritten if existed).\n', file_name);
end
```