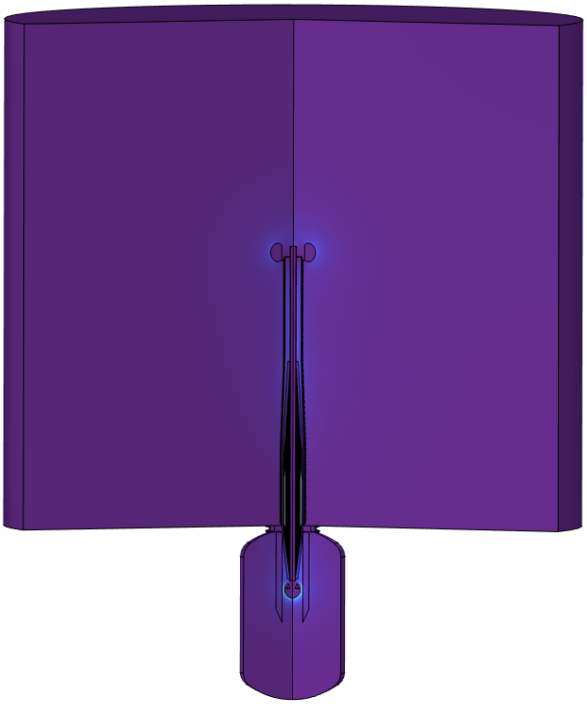


# High Voltage Transformer Bushings: Model To App

Master's thesis in Electric Power Engineering

ARAVIND RAVI

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis In Electric Power Engineering 2025

# High Voltage Transformer Bushings: Model to App

ARAVIND RAVI


CHALMERS
UNIVERSITY OF TECHNOLOGY

*Division of Electric Power Engineering*
Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2025

# High Voltage Transformer Bushings: Model to App



Supervisor: Christos Athanasopoulos, R&D Principal Engineer, Hitachi Energy
Examiner: Prof. Yuriy Serdyuk, Chalmers University of Technology

Master's Thesis 2025
Division of Electric Power Engineering
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone: +46 31 772 1000

Cover:Revolved 2D Axisymmetric Bushing Geometry.

# High Voltage Transformer Bushings: Model to App

**ARAVIND RAVI**
Division of Electric Power Engineering
Department of Electrical Engineering
Chalmers University of Technology

## Abstract

Transformer bushings play a critical role in ensuring the efficient and safe operation of power transformers. Understanding the electrical and thermal behavior of these components is essential for optimizing their design and performance. Therefore the main outline of the thesis is to develop a comprehensive analysis application for transformer bushings using COMSOL Multiphysics software. The application integrates electrical and thermal simulations to provide a holistic understanding of bushing performance under various geometric parameters. The thesis discusses the modelling approach, including the incorporation of complex material properties and boundary conditions. Furthermore, it highlights the transition from the initial computational model to a user friendly application interface, allowing engineers and designers to efficiently analyze and optimize the performance of transformer bushings.

# Acknowledgements

I would like to express my deepest gratitude to my supervisor Christos Athanasopoulos for his invaluable guidance and unwavering support throughout my thesis journey. His profound knowledge and expertise were instrumental in helping me grasp the fundamentals and navigate the complexities of my research. His daily interactions, constructive feedback and insightful suggestions were crucial in shaping the direction of my work. His positive attitude and kind words of encouragement consistently motivate me to overcome challenges and progress with confidence.
I am also profoundly grateful to Francisco Penayo for offering me the oppurtunity to collaborate with the team. The collaborative spirit of the team greatly enriched my research experience. Working alongside such a dedicated and skilled team was both inspiring and rewarding.
Furthermore, I extend my heartfelt thanks to my examiner Yuriy Serdyuk for providing me with the incredible opportunity and for his guidance in exploring the fascinating field of high voltage engineering and COMSOL. His passion for teaching and commitment to student success were evident throughout our interaction and I feel fortunate to have him as my examiner.
Lastly, I want to express my appreciation to all those who contribute to this work, whether through insightful discussions, technical support, or simply by being a source of inspiration. This thesis is a culmination of collective efforts and support of many individuals, and I am deeply grateful to each one of them.

Aravind Ravi, Gothenburg, January 2025

# Contents

# List of Figures

# List of Tables

# 1

# Introduction

Transformer Bushings plays a crucial role in the efficient working of power transformers. Bushings are electrical insulators, supporting the high voltage conductors while providing a barrier between the energized components and the grounded casing. This thesis seeks to explore the design and optimization of transformer bushings paramount for ensuring reliability, minimizing losses and enhancing overall performance and significantly aims to bridge the gap between theoretical modelling and practical application by developing a comprehensive parameterized model of transformer bushing using COMSOL Multiphysics and subsequently transforming it into a user friendly application.

In the dynamic landscape of energy systems, Hitachi Energy stands as a multinational powerhouse dedicated to innovation and excellence. With a legacy spanning over a century, Hitachi Energy has been at the forefront of developing cutting edge equipment for the energy industry. Among its diverse portfolio of services, the manufacturing of transformers plays a pivotal role in the global supply chain. At the heart of every transformer lies a critical component-the bushings. This specialized insulator serves as the vital link between the internal winding's of the transformer and the external transmission system cables. Its role is multifaceted:enabling seamless interconnection, ensuring electrical integrity, and safeguarding against potential failures.

The manufacturing process of bushings demands meticulous care. Operating at high voltages, these connection points generate electromagnetic phenomena that stress the insulation. Any compromise could lead to catastrophic short circuits. Hitachi Energy's engineers approach this challenge with precision, crafted bushings that exceed international safety and reliability standards.

Power transformers are critical components in electrical distribution systems, providing the transmission and distribution of electricity across the vast networks. In recent years,there has been a growing emphasis on improving the efficiency and reliability of power transformers to meet the increasing demands of modern electricity systems. Central to their functionality are transformer bushings, which serve as vital interface between the energized components of the transformer and the external electrical systems. Efficient operation and reliability of transformers heavily rely on the performance of these bushings. This emphasis has spurred significantly advancement in transformer bushing technology leads to the development of more sophisticated bushing models.

## 1.1 Transformer Bushing

Transformer bushings are essential components that provide electrical insulation and support for conductors where they pass through transformer tank walls. They help maintain the integrity of the electrical insulation system,ensuring safe and efficient operation of the transformer. A bushing is a hollow electrical insulator that allows an electrical conductor to pass safely through a conducting barrier such as the case of a transformer without making electrical contact with it. The bushing controls the shape and strength of the field and reduces the electrical stress in the insulating material.
A bushing must be designed to bear extensive intensity of electric field. When the impact of electric field is more, a leakage path is developed inside the insulation. If the energy of the leakage path overcomes the dielectric strength of the insulation, it may puncture the insulation and allow the electrical energy to conduct to the nearest earthed material causing burring and arcing.
Hitachi Energy is a global leader in the field of transformer bushings, has consistently met the diverse needs of the energy industry for over 100 years, the comprehensive portfolio includes both oil-filled and dry bushings, meticulously designed for application spanning from AC to DC. These critical components play an essential role in electrical networks, ensuring safety, reliability and efficient power transmission.

## 1.2 COMSOL Multiphysics

COMSOL Multiphysics is a powerful simulation platform used by engineers and scientists across various fields of engineering, manufacturing and scientific research. It allows you to simulate designs, device and processes by providing fully coupled multiphysics and single-physics modelling capabilities. Whether you,re analyzing structural mechanics, heat transfer, fluid dynamics, electromagnetic, or any other physical phenomenon, COMSOL Multiphysics enables you to understand, predict, and optimize your systems.[2]
The Model Builder includes all of the steps in the modelling workflow-from defining geometries, material properties, and the physics that describes specific phenomena to performing computations and evaluating the results. When the model is developed the Application Builder is used to turn it into simulation application with a dedicated user interface that can be used by collaborated and customers who are not experts in the simulation software.[2]

## 1.3 Thesis Objectives

In the realm of engineering and testing, simulation serves as indispensable tools for assessing performance,predicting behaviour, and optimizing designs, Before venturing into the real world experiments, it is essential to simulate the intended scenarios.However, a fundamental prerequisite for simulations is having a well defined object to work with.

Bushings geometry play a pivotal role in electrical integrity and safety. Creating these models involved manual effort, consuming valuable time and resources. Instead of manually constructing bushing geometries,this thesis proposes leveraging the power of Application builder in COMSOL Multiphysics. By parameterize the bushing model and transforming it into a user friendly application. The objective is to provide design engineers with efficient tools for optimizing bushing designs, contributing to more accessible engineering solutions.
Benefits of the new approach are, By automating geometry creation reduces the need for painstaking manual work. Engineers can focus on higher level tasks.Hence time can be saved. The application builder ensure consistent geometries across different simulations. Parameterization allows easy adjustments, accommodating varying design requirements. Running specific methods generates geometries swiftly, enhancing productivity. The integration of COMSOL Multiphysics is seen as a way innovation in this domain.

# 2

# Theoretical Background

In this chapter, we delve into the fascinating realms of electrostatics and heat transfer, meticulously analyzing their significance in simulating parameterized bushing models.

## 2.1 Electrostatics

Electrostatics deals with the study of electric charge at rest. We study electric field, potentials and charges but ignore currents and magnetism which may also be present.

### 2.1.1 Electrostatics in free space

Electrostatics deals with describing an electric field caused by static charges. The electric field concept arises from the interaction between charges. Imagine one charge creating an electric field throughout space. When we introduce another charge into the field, the force acting on it is determined by the electric field at its location. Starting with free space, assuming a space charge density $\rho$, the relationship with the electric field (E)is [5].

$$\nabla \cdot E = \rho/\epsilon_0 \tag{2.1}$$

where $\epsilon 0$ is a universal constant of nature called permittivity of free space.
The space charge density acts like a volume source. The Maxwell's equations implies electric field is irrotational (curly free), which is the static version of Faraday's law.

$$\nabla \times E = 0 \tag{2.2}$$

for irrotational field, there is scalar potential which leads to electric potential V;

$$-\nabla V = E \tag{2.3}$$

The vector identity is ;

$$\nabla \times \nabla V = 0 \tag{2.4}$$

Hence combining the Maxwell's equation for electrostatic the equation is written as;

$$-\nabla \cdot \nabla V = \rho/\epsilon_0 \tag{2.5}$$

Even though the use of this equation is limited due to inability to represent the dielectric materials.

### 2.1.2 Electrostatics in dielectric

An ideal dielectric lacks free charges, but it contains bound charges within its structure.When an external electric field is applied, these bound charges can shift,leading to the formation of induced electric dipoles. Each induced dipole consists of a positive charge and negative charge and they align themselves with the direction of the electric field.

As a result, the overall electric field inside a dielectric material differs from that in free space.Specifically, the presence of induced dipoles modifies the electric field lines, making them denser within the dielectric. This phenomenon affects capacitance, energy storage and other electrical properties when dielectric are used in capacitor or other devices.

To get more description we need to introduce a polarization vector field P, and a polarization charge density, $\rho p$. The polarization effect inside the dielectric can be modified as [5].

$$\nabla \cdot E = (\rho + \rho p)/\epsilon_0 \tag{2.6}$$

Based on this the electric displacement field D can be defined as,

$$D = \epsilon_0 E + P \tag{2.7}$$

Hence the electrostatics equation can be defined as,

$$\nabla \cdot D = \rho \tag{2.8}$$

Based on the above equations we can combine it together as,

$$-\nabla \cdot (\epsilon_0 \nabla V - P) = \rho \tag{2.9}$$

### 2.1.3 Electrostatic Equations and Boundary Conditions for Material Interface

Gauss's law relates the electric flux through a closed surface to the enclosed charge, while Faraday's law describes electromagnetic induction. According to Helmhotz's theorem, these law determine the electric field upto an unknown constant. This constant underscores the need to specify a ground level of electric potential. At material interface(where different material meet), the divergence condition pertains to the normal component of the field, and the curl condition pertains to the tangential component. These conditions help us understand how to impose appropriate boundary conditions. The integral forms of these laws often used to derive boundary formulations by considering shrinking closed surfaces(for Gauss's law) and contours(for Faraday's law)that enclose portions of material interfaces[5].

**Table 2.1:** Summarized conditions in laws

| Equation Name | Differential Form | Integral Form | Boundary Condition |
|---|---|---|---|
| Gauss's law | All field lines start and end on charges | The total flux through a closed surface equals its enclosed charge | The surface charge at a material interface equals the jump in the normal component of the displacement field. |
| Faraday's law | The electric field of irrotational | The electric field is conservative | Across a material interface, the tangential component of the electric field is continuous. |

## 2.2 Heat Transfer

It is essential to investigate the thermal characteristics of transformer bushings because unusual increase in temperature can lead to localized insulation failures, which in turn can significantly compromise the reliability and safety of the electrical power system. The primary sources of heat in bushings are the joule heating from the central conductor and the eddy current losses in the components such as the flange and resin-impregnated paper. This generated heat is then dispersed through the convection of air and oil, as well as the conduction between the insulating material and the wall bushing [3].

In a bushing, heat production primarily arises from two phenomena:Joule heating and eddy current loss heating. During standard operation, the bushing's central conductor carries alternating current, leading to joule heating. Concurrently, the alternating electromagnetic field prompts the formation of eddy currents within the bushing's flange.The resin impregnated paper, and other components, resulting in heat generation due to eddy current losses. The bushing's heat dissipation involves two key mechanisms:conduction and convection. As depicted in figure 3.1, the heat dissipation pathway [3].

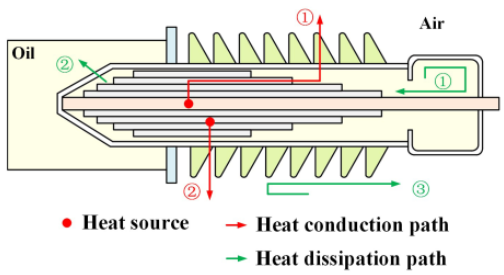



**Figure 2.1:** Schematic diagram of heat transfer in bushing [3]

### 2.2.1 Conduction, Convection and Radiation

#### 2.2.1.1 Conduction

The process of heat conduction can be encapsulated by the equation;

$$q = -\lambda \nabla T \tag{2.10}$$

q represents the heat flux density, measured in watts per square meter($W/m^2$). $\lambda$ symbolizes the thermal conductivity of the material, with units of watts per meter/kelvin (W/(m.K)). $\nabla$ signifies the vector differential operator. T denotes the temperature of the material, in kelvins(K), across the section being analyzed.
This equation (3.1) indicates that the thermal conductivity of the material is a crucial factor in the effectiveness of heat transfer by conduction. The distribution of temperature within the material is determining factor for the thermal conductivity, which in turn, influences the overall temperature distribution within the bushing.Therefore, accurately setting thermal conductivity is essential when creating a simulation model to predict heat distribution accurately [3].

#### 2.2.1.2 Convection

The thermal Convection process consists of two primary components: Forced convection heat transfer and natural convection heat transfer.
1)Natural convection of oil in bushing: The bushing is a component that connects the transformer tank to external equipment. It allows electrical conductors to pass through the tank while maintaining insulation. At the lower end of the bushing(inside the transformer tank), the oil temperature is close to the transformer's top oil temperature. Due to the low density of the oil in this area, it tends to rise. Conversely, at the upper end of the bushing(exposed to air), the oil temperature is closer to the ambient temperature, resulting in higher density. Cold oil tends to sink. This density difference sets up a natural circulation flow within the bushings. Hot oil rises, and cold oil descends due to gravity. The oil flow enhances heat diffusion within the bushing, and its flow rate significantly affects the steady-state temperature distribution in the bushings [3].
2)Natural Convection of oil in the tank: When the bushing operates under rated current, the lower end of bushing typically has higher temperature than the transformer's top oil temperature. Oil flow within the bushing contributes to heat diffusion. However, the temperature difference between the lower end and the bottom of the bushing is relatively small. Consequently, the heat dissipated from this path does not play a dominant role in overall heat transfer [3].
3)Natural Convection of Air: Most of the bushing's heat dissipates to the environment through convective heat transfer between the insulator(part of bushing)and the surrounding air. Air flow rates are much higher than those of oil. Therefore, simulations often use surface heat flux boundary conditions to model the ideal convective heat transfer with air [3].
The thermal convection process can be described as;

$$q = h\Delta T \tag{2.11}$$

where q is the heat flow density ($W/m^2$). h is the convective heat transfer coefficient ($W/m^2.K$). $\Delta T$(K) is the temperature difference between the two sides of the contact surface.
Joules heating of the center conductor and eddy current loss heating of the resin impregnated paper and other structures. Heat transfer modes experiences both heat convection and conduction here, whereas heat dissipation is poor at the bottom end.

#### 2.2.1.3 Radiation

Radiation is a significant factor in the operating environment, with its intensity varying due to factors like wind, speed, temperature, and weather conditions. Thermal Radiation primarily occurs between the outer surface of bushing and the transformer block. The temperature of the transformer bushing increase due to current heating and dielectric loss, leading to heat transfer between the bushing and the transformer body. Additionally, the sun contributes to heat transfer by radiating heat outward based on its temperature, forming a system where heat is exchanged through radiation between the transformer bushing, body and the surrounding environment[4].

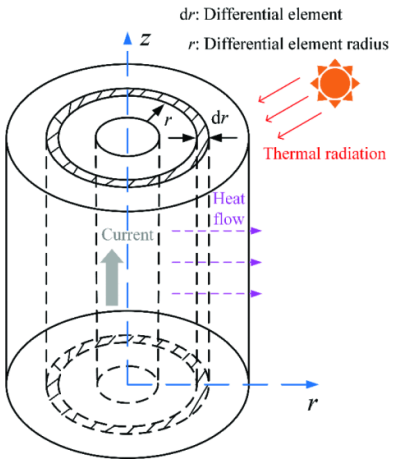



**Figure 2.2:** Schematic diagram of heat source and flow direction of bushing [4]

# 3

# Methods

The complex behaviour of bushings poses challenges in accurately representing their dynamics with computational models. This research endeavours to develop a robust methodology that only captures the intricacies of bushing behaviour but also facilitates their integration into simulation frameworks for practical applications.

The initial step involve constructing the model in COMSOL, starting with the creation of an axissymmetric geometry to represent the bushing components. This includes delineating various parts such as conductor, condenser core, flange, insulator, sheds, testing tanks, corona and foils. Using the Model builder, we follows a structured approach to generate the geometry. Two components are created to simulate electrostatics and heat transfer physics. Both components are meshed using a physics-controlled approach. A stationary study is performed for the electrostatics component, while a time-dependent study is conducted for the heat transfer physics. After simulating and analyzing the model using the Application Builder, a parameterized model is developed. This leads to the creation of a user-friendly application designed to verify various geometric conditions efficiently.

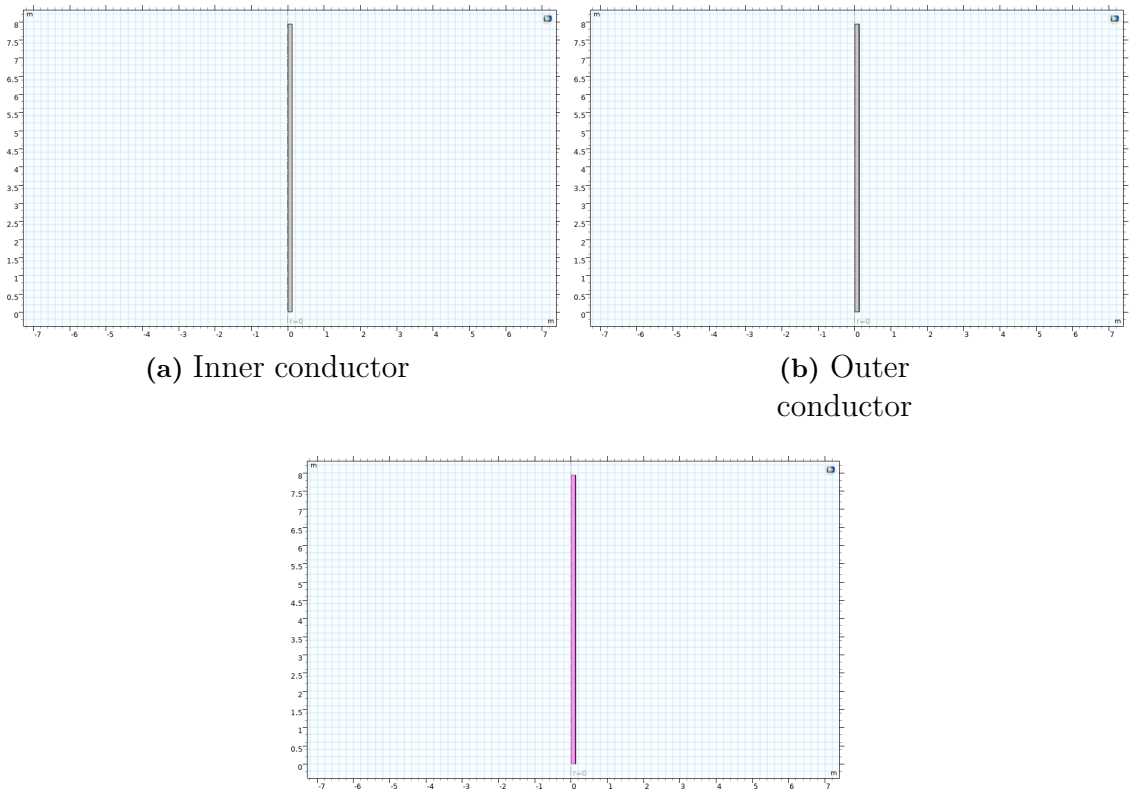
## 3.1 Model Builder

### 3.1.1 Geometry

Creating the geometry using Model Builder involves several steps to ensure each component is accurately represented and can be separately selected. Separate geometries are created for both the electrostatics and heat transfer physics components. The geometry for the heat transfer physics specifically includes only the conductor, condenser core, flange, insulator, and sheds. Here's a systematic breakdown:

#### 3.1.1.1 Conductor

The inner and outer conductor are created using 2D rectangle primitives. The difference between the inner and outer conductors is achieved using the Boolean and partition operation, allowing for the separate selection of each geometry. In the figure 3.1 below shows how the geometry is created for (a)Inner conductor, (b)Outer Conductor, (c) Difference between inner and outer conductor.



**(a)** Inner conductor

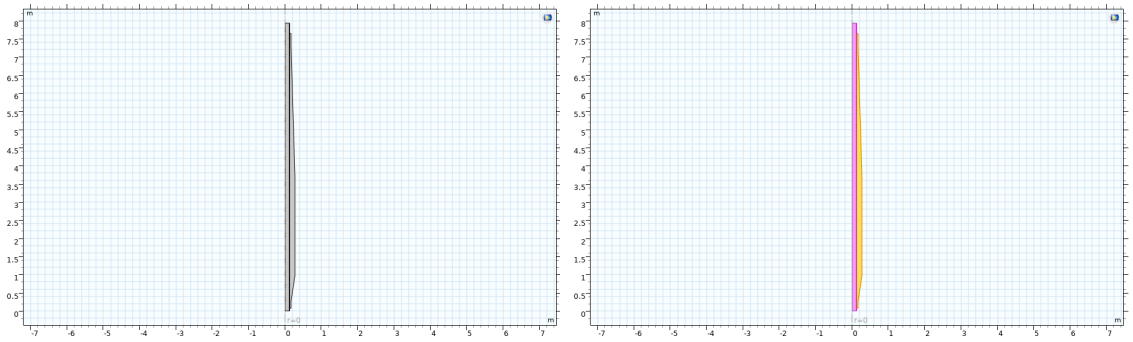
**(b)** Outer conductor

**(c)** Difference

**Figure 3.1:** Conductor Geometry

#### 3.1.1.2 Condenser Core

The condenser core is constructed using polygon primitives. To facilitate the separate selection of each geometry, the difference option is included using the Boolean and partition operation. Figure 3.2 shows how the condenser core geometry looks like (a)Condenser core, (b)Difference between condenser core and conductor



**(a)** Condenser core

**(b)** Difference

**Figure 3.2:** Condenser core Geometry

#### 3.1.1.3 Flange

Polygon primitives are used for creating flange. Figure 3.3 shows how the flange geometry is build.

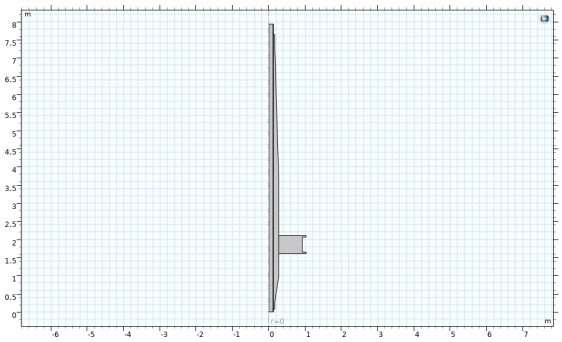
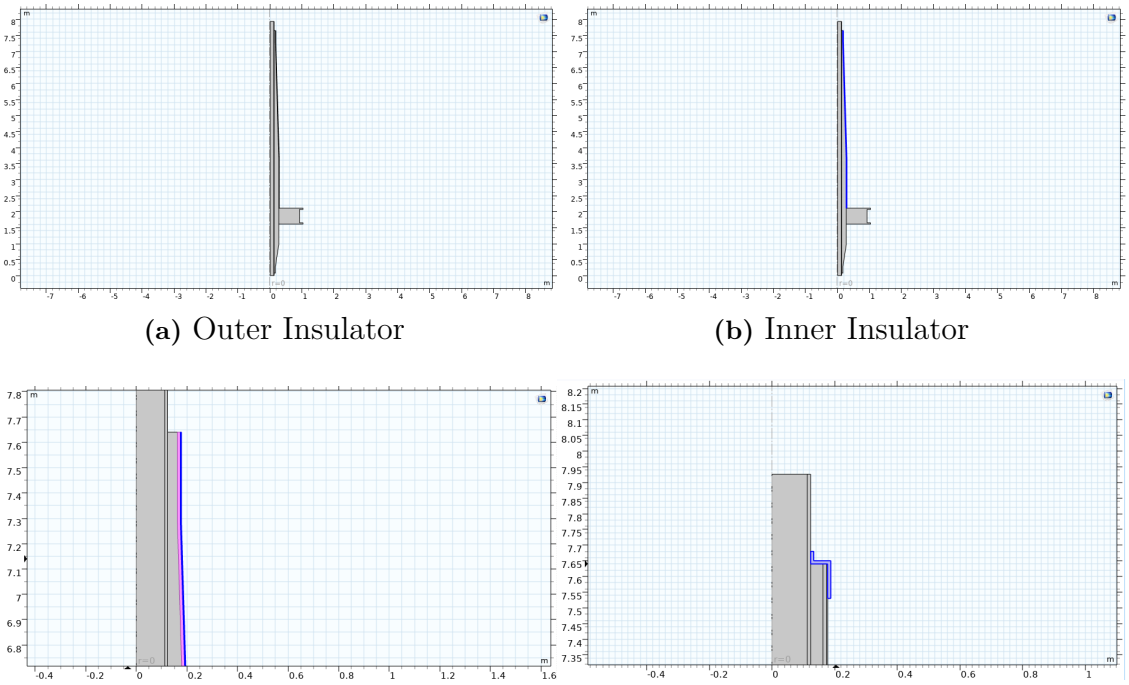


**Figure 3.3:** Flange Geometry

#### 3.1.1.4 Insulator and Top piece

The inner and outer insulators are made using polygon primitives. The difference is added to both insulators using the Boolean and partition operator, enabling separate selection.Top piece is created using polygon primitives. Figure 3.4 consist of (a) Outer insulator, (b) Inner insulator, (c) Difference between inner and outer insulator,(d) Top piece geometry.



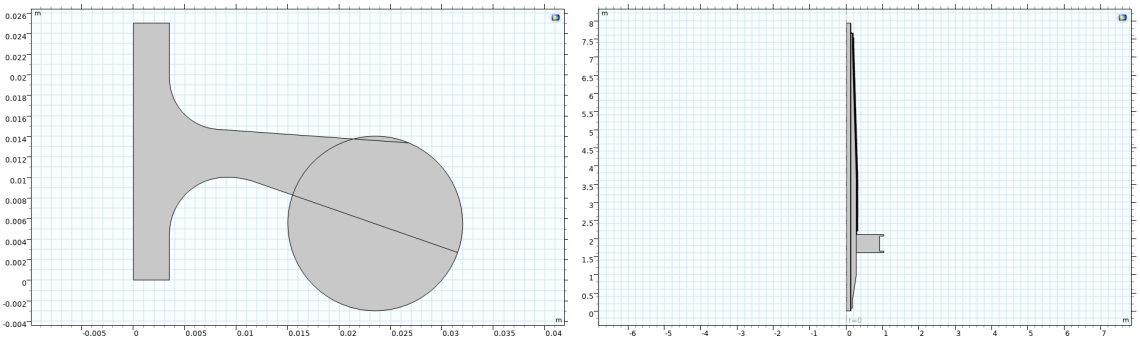
**(a)** Outer Insulator **(b)** Inner Insulator

**(c)** Difference **(d)** Top piece

**Figure 3.4:** Insulator Geometry

#### 3.1.1.5 Shed

Sheds are designed as a geometry part using polygons, circles and fillets, and then integrated into the main geometry. Figure 3.5 consist of (a) Shed geometry, (b) How sheds are added in the main geometry.



**(a)** Shed Geometry

**(b)** Sheds in main geometry

**Figure 3.5:** Shed Geometry

#### 3.1.1.6 Tank and Corona

The test tank and corona are imported as CAD files and inserted into the geometry. Figure 3.6 represents tank and corona in main geometry.

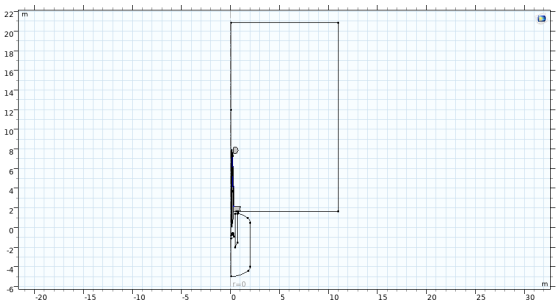


**Figure 3.6:** Tank and Corona Geometry

#### 3.1.1.7 Foils

Foils are included in the condenser core for capacitive grading to ensure the electric field is evenly distributed across the core. Given the practical difficulty of creating each of the approximately 124 foils individually, an Excel database is used to represent full and partial foils. Java code,leveraging Excel Live link,imports the foil data and plots it in the condenser core. Figure 3.7 shows how the foils are included in the geometry.

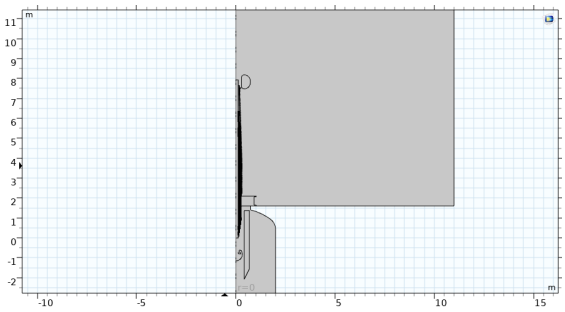


**Figure 3.7:** Foils in Geometry

### 3.1.2 Materials

Once the geometry is build, The next step is to assign appropriate materials to each geometric domain to accurately represent the physical properties of the system.To do this, navigate to the 'Materials' node in the Model Builder tree and select 'Add Materials'. From the material library,choose the specific material that corresponds to each part of geometry. After selecting a material, ensure to define all necessary material properties, such as density,thermal conductivity, electrical conductivity, and any other relevant physical attributes required for your simulation. If the material needed is not available in the library either it can be created manually by adding properties or can be imported. Assigning materials accurately is crucial as it directly affects the simulation results. Each geometry part must be assigned its respective material properties to reflect real world behavior. After assigning materials, verify that each domain is correctly linked to its material to avoid any discrepancies during simulation. The material to the bushing geometry can be assigned as follows;

#### 3.1.2.1 Air

Air is defined for the inner conductor and surroundings for the test. As in the figure 3.8;

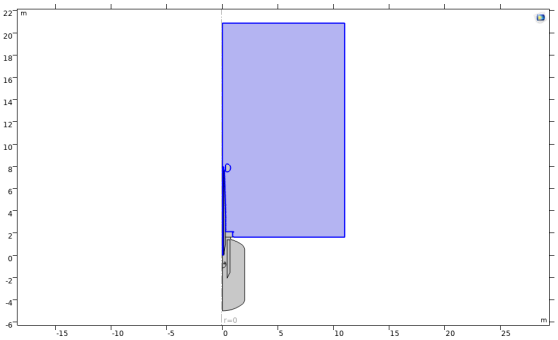


**Figure 3.8:** Air

#### 3.1.2.2 Copper

Conductor is defined as copper in the figure 3.9.

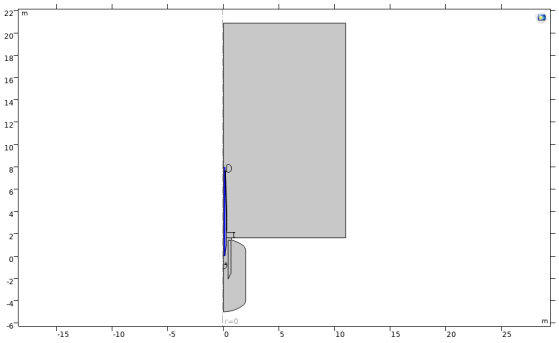


**Figure 3.9:** Conductor

#### 3.1.2.3 Resin impregnated paper

Condenser core is made up of resin impregnated paper where foils are included inside the condenser core. The figure 3.10 define the material in the condenser core;

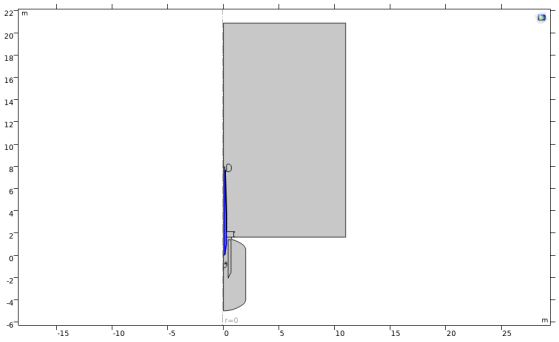


**Figure 3.10:** Condenser core

#### 3.1.2.4 Gel

Gel is defined as the material for the insulator as in the figure 3.11;

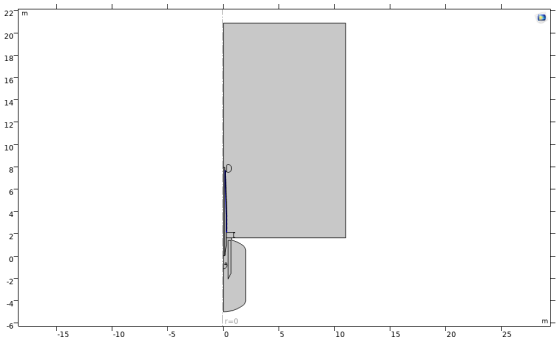


**Figure 3.11:** Insulator

#### 3.1.2.5 Composite Silicon Resin

Sheds are made up of composite silicon resin as in below figure 3.12;

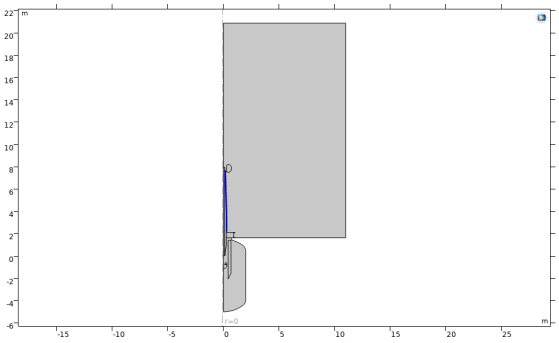


**Figure 3.12:** Sheds

#### 3.1.2.6 Aluminium

Top piece and Flange are defined as aluminium as int he figure 3.13 below;

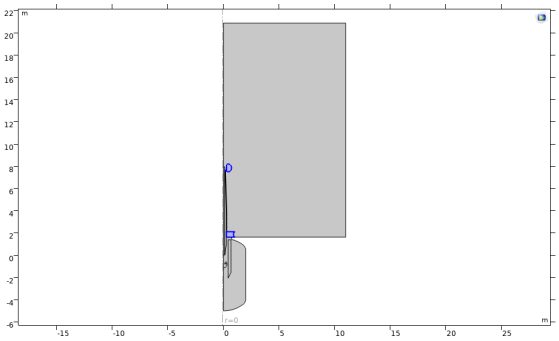


**Figure 3.13:** Flange and Top piece

#### 3.1.2.7 Transformer Oil

Test tank is considered to have transformer oil as in the figure 3.14 below;

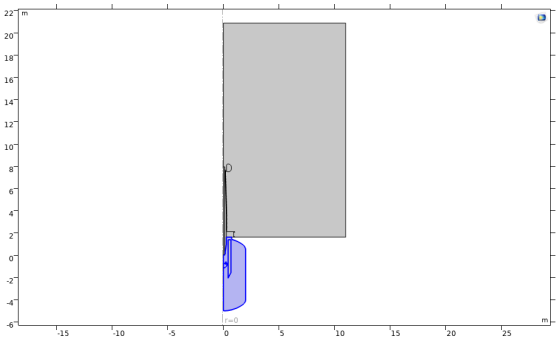


**Figure 3.14:** Tank

### 3.1.3 Boundary Conditions

Setting boundary condition is crucial for accurate simulations in both electrostatics and heat transfer modules. For electrostatics, boundary conditions typically includes specifying electric potential on the boundary domains. Grounding is applied to

reflect the physical setup of the problem. In the heat transfer module, boundary condition often involve setting temperature values, applying convection heat transfer coefficients to model to know the exchange of heat with the surroundings. Accurate definition of these boundary conditions ensure that the physical behaviour of the system is correctly modeled, leading to reliable simulation results.

#### 3.1.3.1 Electrostatics

**Charge Conservation**
Boundary condition for Charge conservation is illustrated as in the figure below 3.15;

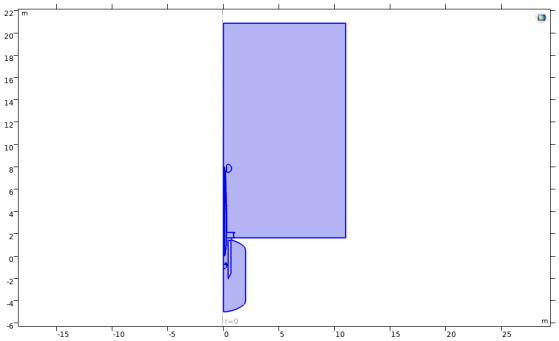


**Figure 3.15:** Charge Conversion

**Axial Symmetry**
Boundary condition for axial Symmetry is represented as in the figure below 3.16;

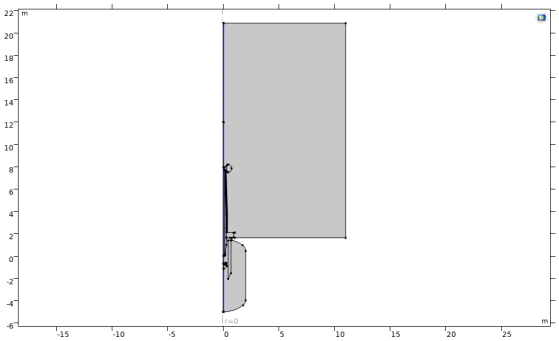


**Figure 3.16:** Axial symmetry

**Initial Values**
Initial value for boundary condition for electrostatics as following figure 3.17;

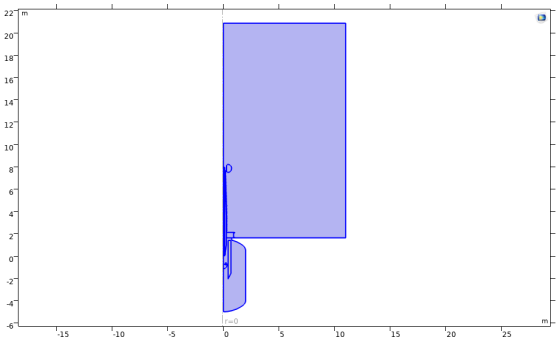


**Figure 3.17:** Initial values

**Electric Potential**
Electric potential is defined for the conductor in the below figure 3.18.

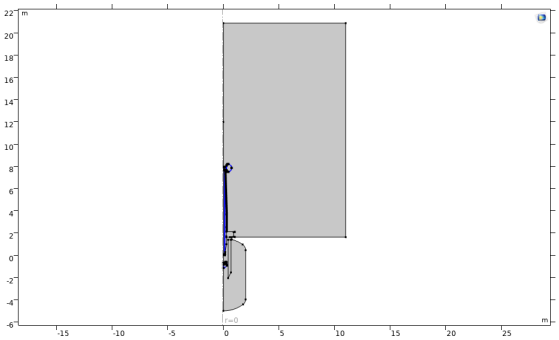


**Figure 3.18:** Electric Potential

**Ground**
Ground is represented as outer portion of the test tank and the surrounding area as in the figure 3.19;

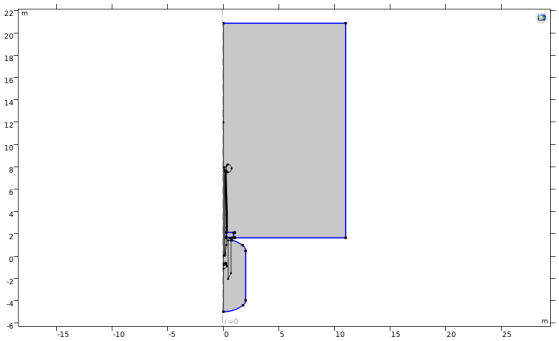


**Figure 3.19:** Ground

#### 3.1.3.2 Heat Transfer

**Solid**
Solid Boundaries are defined as in the figure 3.20 below;

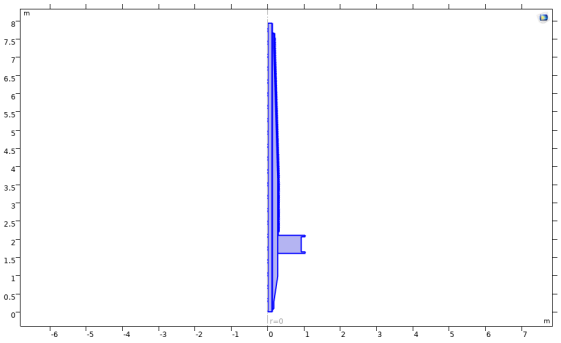


**Figure 3.20:** Solid

### Initial Values

Initial values for the heat transfer boundaries are given as in the following figure 3.21;

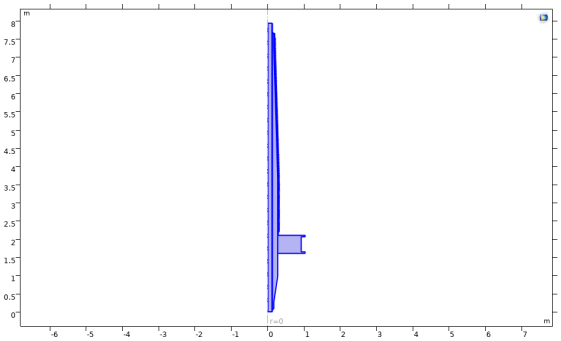


**Figure 3.21:** Initial Values

### Axial Symmetry

Boundary condition for axial Symmetry is represented as in the figure 3.22 below;

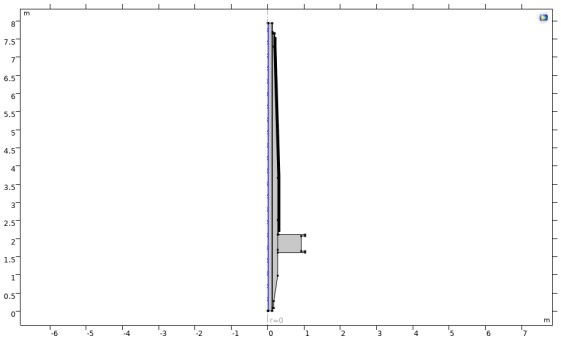


**Figure 3.22:** Axial Symmetry

### Thermal Insulation

Thermal insulation boundaries are represented as in the following figure 3.23;

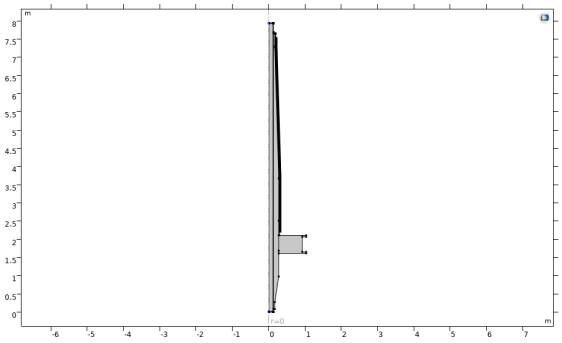


**Figure 3.23:** Thermal Insulation

### Heat Source

Heat source boundaries are represented as in the following figure 3.24;

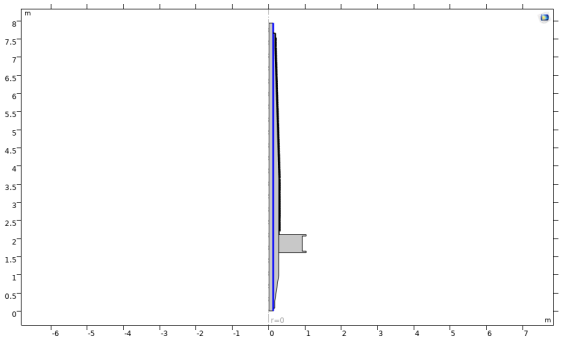


**Figure 3.24:** Heat Source

### Heat Flux Air

Heat flux for air is represented as in the following figure 3.25;

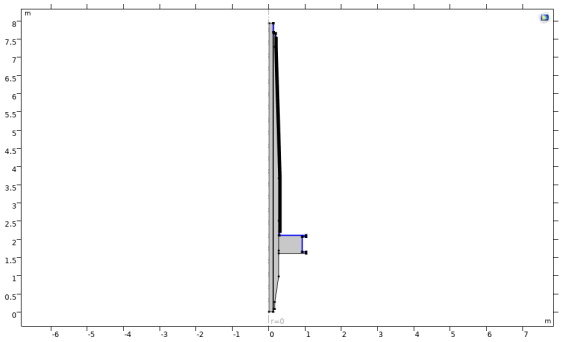


**Figure 3.25:** Heat Flux Air

### Heat Flux Oil

Heat flux for the oil is represented as in the following figure 3.26;

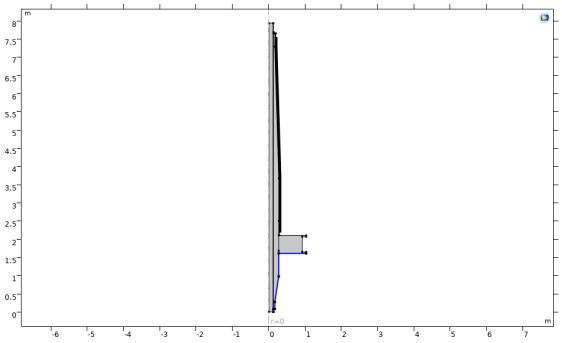


**Figure 3.26:** Heat Flux Oil

**Surface to Ambient Radiation SiR**
Surface to Ambient radiation for the shed (SiR) is represented as in the figure 3.27;

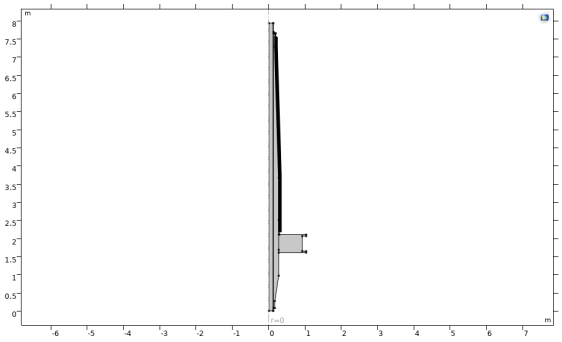


**Figure 3.27:** Surface to Ambient Radiation SiR

**Surface to Ambient Radiation Solid**
Surface to Ambient radiation for solid is represented as in the figure 3.28;

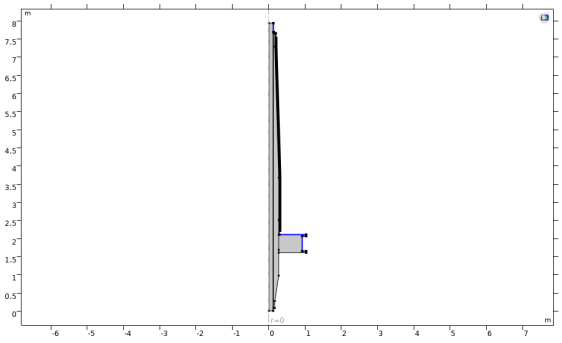


**Figure 3.28:** Surface to Ambient Radiation Solid

## 3.2 Mesh

Meshing is critical step in finite element analysis as it affects the accuracy and computational efficiency of the simulation. Physics controlled mesh ensures that the mesh is tailored to the specific physics and properties of the model leading to

more accurate results. Right click on the mesh node and select physics controlled mesh. Choose the element size of meshing.

### 3.2.1 Electrostatics

The figure 3.29 below represents the physics controlled meshing for electrostatics physics;

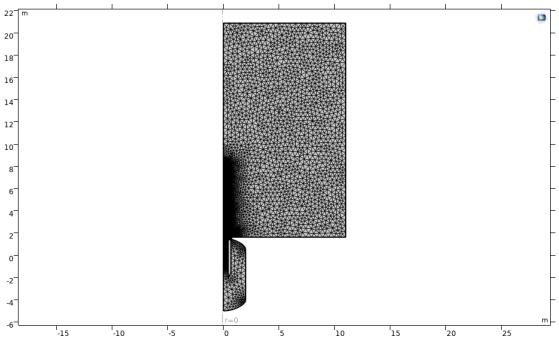


**Figure 3.29:** Electrostatics

### 3.2.2 Heat Transfer

The figure below 3.30 represents the physics controlled meshing for Heat transfer physics;

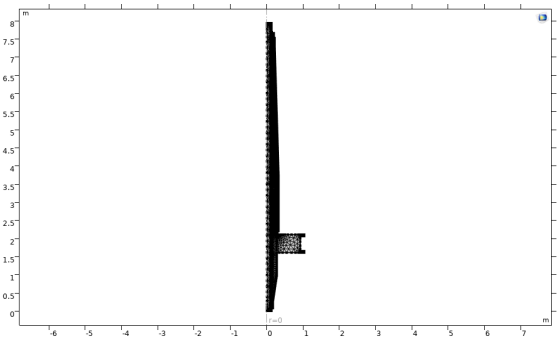


**Figure 3.30:** Heat Transfer

## 3.3 Study Definition

### 3.3.1 Electrostatics

A stationary study for electrostatics is used to analyze and solve problems where the electric field and potential distribution does not change over time. The study is essential for understanding the behaviour of electric fields in static condition. Maximum electric field is calculated over sheds and corona.

### 3.3.2 Heat Transfer

Time dependent study for heat transfer involves simulating how temperature distribution evolves over time. Temperature rise test is done by plotting the cut line over the outer surface of the conductor and finding the hot spot over the conductor with respect to time.

## 3.4 Application Builder

To create a user friendly application in COMSOL, Application Builder is utilized.This tool allows to design the user interface, known as a form, which serves as the application's main interactive component.The Application Builder provides two essential tools for developing the application: The Form Editor and Method Editor. The application can feature a menu or ribbon additionally to enhance navigation.

**Form Editor**
The Form Editor offers a drag and drop interface,enabling easy access to add and arrange user interface components such as input fields, graphic objects, and buttons. The intuitive environment simplifies the process of building and customizing the application's user interface.[6]

**Method Editor**
The Method Editor is powerful programming environment where the data structures representing different part of model can be modified. It allows for advanced customization and functionality, enabling you to tailor the application to specific needs and workflow.[6]

### 3.4.1 Parameters

In a model tree, Parameters are essential for controlling various settings of the model. Parameters are user-defined constant scalars and can be found under the Global Definitions node in the model tree. These parameters are accessible throughout the entire Model Builder, offering a consistent and flexible way to manage model settings.
Parameters can be used to define the coordinates of geometric entities in the Model Builder. By setting up these coordinates as parameters, the flexibility to easily modify the geometry without manually changing each coordinate value. This parametrization streamlines the process of adjusting the model, ensuring consistency and reducing the potential for errors.

#### 3.4.1.1 Integration with Application Builder

The real power of using parameters comes into play when these parameters are used within the Application Builder. Here's how this integration benefits the modelling process.[6]

**1. Input fields for User Interaction**
In Application Builder, creating input fields linked to the parameters that define geometry coordinates. This setup allow user of the application to input or modify values directly, making the application interactive and user friendly.[6]

**2. Parametrizing the Model**
By exposing these parameters as in the input fields in the application, users can easily parameterize the model, Hence the geometric dimensions can be adjusted without needing deep technical knowledge of the underlying model structure. For instance,an engineer could tweak the length, width, or height of a component simply by entering new values into the provided fields.[6]

**3. Dynamic Model Adjustments**
Changes made in the Application Builder automatically get updated for corresponding parameters in the Model Builder. This ensures that the geometry and other aspects of the model are updated in the real time, providing the immediate visual and analytical feedback.[6]

**4. Enhancing the Customisation and control**
User can explore different design scenarios, optimize parameters for better performance, and conduct sensitivity analyses, all within a single integrated environment.[6]

### 3.4.2 Application User Interface

#### 3.4.2.1 Creating New Form

To create a new user interface layout, click the 'New Form' button in the Home tab. This will open the New Form wizard, which assists in adding the most common user interface components. The wizard includes three main tabs for incorporating these interface components as follows;

**Input/Output**
The Input/Output tab showcases nodes in the model tree that can be used as input fields, data display objects, check boxes or combo boxes. The text label and unit can automatically included with input fields. Additionally, other components of the model can be integrated using the Model Data Access feature. This allows foe a comprehensive customization of the interface by incorporating various model elements seamlessly.[6]

**Graphics**
The Graphics tab represents nodes from the model tree that can be utilized as graphics objects. These include Geometry, Selection Mesh and Results. By using the tab,you can add and customize visual elements such as geometric shapes, selection meshes, and result visualizations, enhancing the graphical representation of the model. This feature helps for detailed and dynamic graphical interface, providing clear and informative visual feedback with the user interface.[6]

**Buttons**
The Button tab displays nodes from the model trees that can be executed by clicking button in the application's user interface. Action such ad Plot Geometry, Plot Mesh, and Compute Study can be initiated through these buttons. Additionally using the Method Editor, you can incorporate custom code into the buttons, enabling the execution of tailored scripts and functions. This feature provides a powerful way to enhance interactivity and functionality within your application, allowing users to perform complex operations with a single click.[6]

### 3.4.3 Application Features

The user interface is organized into three tabs: Instruction, Electrostatics, Heat Transfer

#### 3.4.3.1 Instruction Tab

The instruction tab provides an explanation of how the application functions and outlines the procedures necessary to operate it. This section includes step by step guidance on setting up and using the application, ensuring that user can easily understand and follow the required steps to achieve their objectives. It serves as a comprehensive manual, offering clear instructions and tips for optimal usage.

#### 3.4.3.2 Electrostatics Tab

The Electrostatics tab is further divided into two sub tabs: Geometry and Compute

**Geometry Tab**
This tab contains the components used within the bushing and assist in the parametrization of the bushing geometry.

**Conductor**
The user interface of the conductor is designed as follows in the figure 3.31;

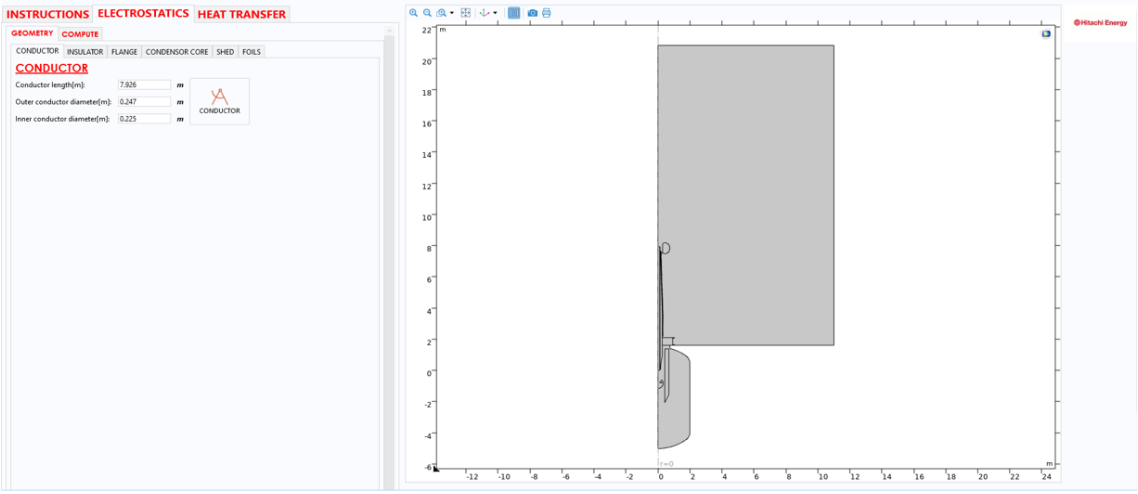



**Figure 3.31:** User interface of the conductor

The user interface for the conductor includes input fields for specifying the conductor length, outer conductor diameter and inner conductor diameter. These parameters are integrated into the application from the model builder, allowing user to define and adjust the conductors dimensions. When the conductor button is click, it plots the corresponding changes in the conductor within the graphics window.

### Condenser Core

The user interface for the condenser core is as follows in the figure 3.32;

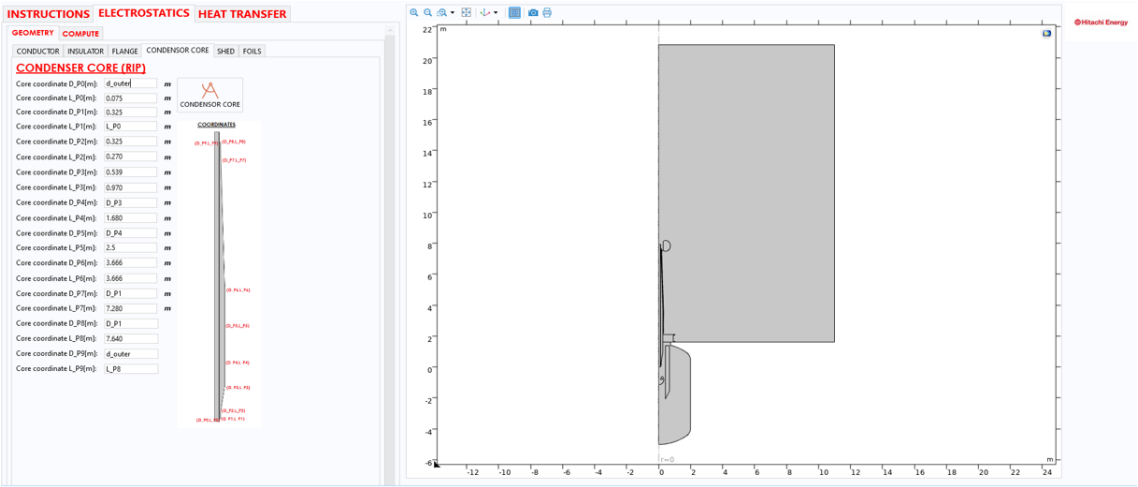



**Figure 3.32:** User interface for condenser core

The user interface for the condenser core includes fields for entering the diameter and length coordinates. These coordinates are visually represented with a figure to facilitate easy understanding.By clicking the Condenser core button, user can plot the corresponding changes in the condenser core within the graphic window. This functionality allows for immediate visual feedback, helping user to accurately adjust and visualize the dimension of the condenser core in the real time.

### Flange

The user interface for flange geometry is designed as following figure 3.33;

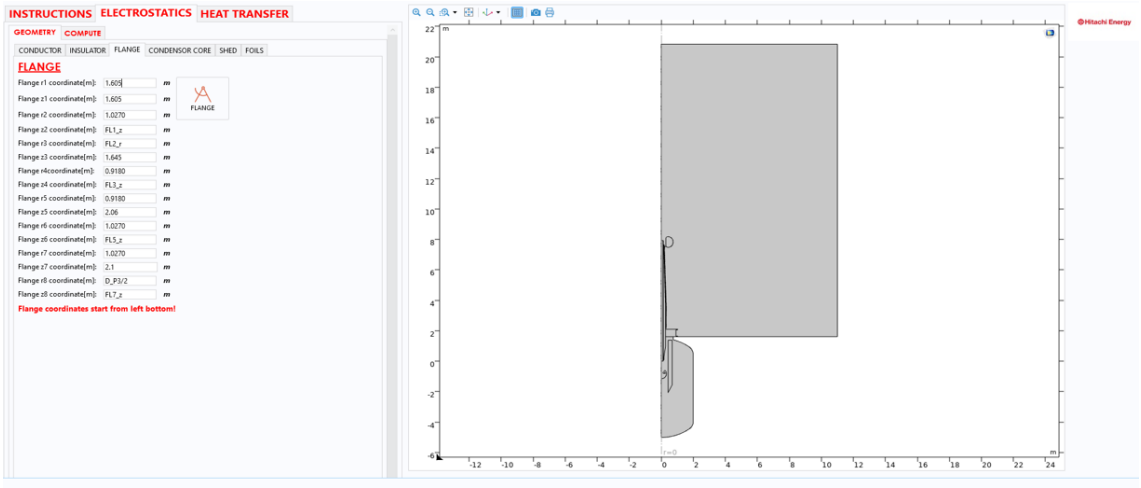



**Figure 3.33:** User interface for Flange

The user interface for the flange include fields for entering its coordinates ,which start from the left bottom of the geometry. User can input these coordinates to define the position and dimensions of the flange. When the flange button is clicked, it plots the corresponding changes in the flange within the graphics window.

### Insulator and Top piece

The user interface for the insulator and Top piece is designed as follows in the figure 3.34;

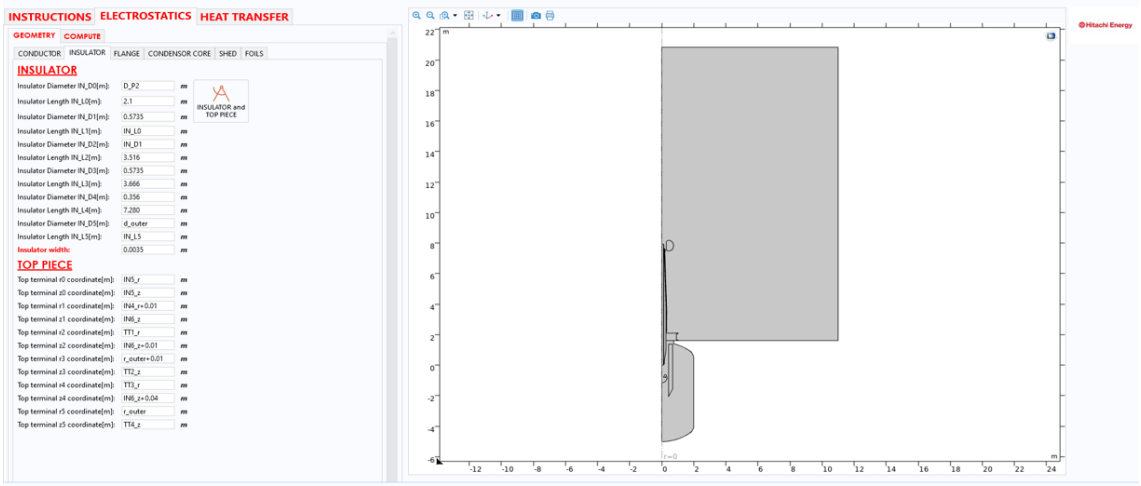



**Figure 3.34:** User Interface for Insulator and Top piece

The user interface for the insulator allows user to enter the diameter and length coordinates, as well as adjust the width of the insulator.Additionally, there are fields for specifying the coordinates of the top piece to modify its geometry. By clicking the Insulator and Top piece button, user can plot the corresponding changes in both the insulator and the top piece within the graphic window. This help in ensuring that changes are accurately reflected in the overall design.

### Shed

The user interface for the sheds are designed as follows in the figure 3.35;

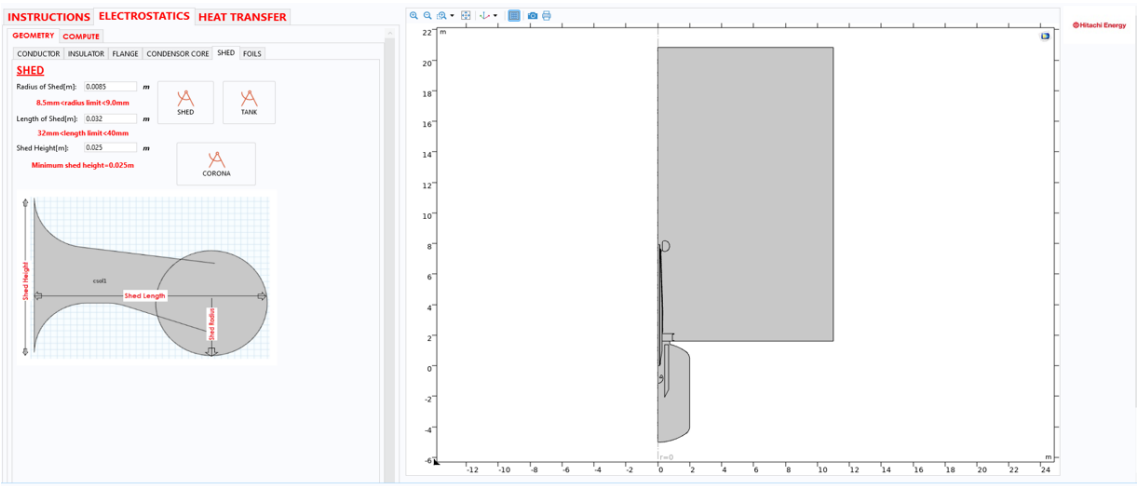



**Figure 3.35:** User interface for sheds

The user interface for the shed allows user to adjust the radius, length and height of the shed. The radius can be set between 8.5mm and 9.0mm, the length can be adjusted between 32mm and 40mm, and the maximum height is capped at 0.025m. These parameters are visually represented in a figure within the interface for easy reference.
By clicking the Shed button, user can plot the changes in the shed parameters in the graphics window. Additionally, there are buttons for the tank and corona, which can be used to plot their respective changes in the graphics window as well.

### Foils

The user interface for the foils are designed as follows in the figure 3.36;

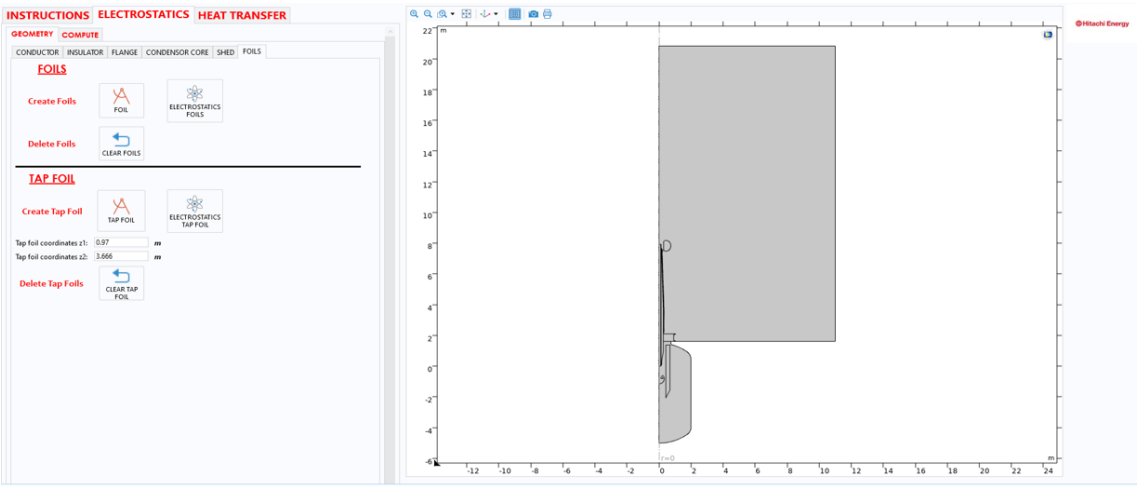



**Figure 3.36:** User interface for foils

To incorporate foils into the geometry, a database is created in an Excel file, detailing

the number of foils, their dimensions and whether they are partial or full foils. By using the Excel Live-link and method editor in COMSOL, a code is generated to import this data into COMSOL. When the Foil button is clicked, the foil geometry is generated and displayed in the graphic window.
The Electrostatic button is used to apply boundary conditions to the entire geometry, including the foils. A java code is implemented to allow for deletion of the generated foil geometry and to facilitate cumulative selection of the geometry.
For creating tap foil inside the condenser core, specific coordinates for the tap foil are provided. A dedicated button is available for generating only the tap foils. The boundary conditions for the tap foil and the entire geometry are set using the Electrostatics Tap Foil button. To clear the tap foil, a Clear Tap Foil button is also provided. These buttons are formatted and fictionalized using java code in Method Editor.
Additionally different bushing foils can be imported into the application by changing the foil name in the java code, allowing for easy customization and flexibility in the design process. The detailed code for implementing the buttons can be found in the appendix section.

### Compute Tab

The tab is designed for meshing and computing the geometry. The tab look like as follows in the figure 3.37;

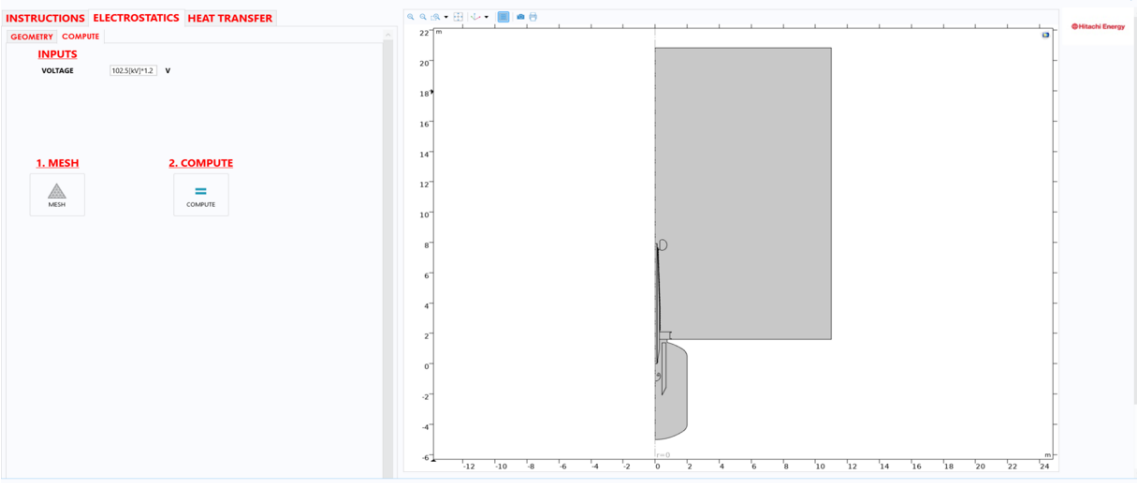



**Figure 3.37:** User interface of Compute Tab

The compute tab within the electrostatics module is specifically designed for meshing the geometry ans simulating the physics. This tab includes critical components such as the input voltage field, meshing button and compute button.
The input voltage field allows the user to specify the voltage parameters necessary for the simulation. The input is crucial as it defines the electrical conditions applied to the geometric model.
The meshing section provides tools for creating a mesh that represent the geometry. A physics controlled mesh is automatically applied to the geometry, ensuring

that the mesh density and distribution are optimized for accurately capturing the electrostatics phenomenon.
The compute button is the final step in the process. By clicking this button, you initiate the simulation,which then uses the specified voltage and the generated mesh to solve the electrostatics equations. This process yields the simulation results, which can be analyzed to understand the behaviour of the system under the given condition.

#### 3.4.3.3 Heat Transfer Tab

The Heat transfer tab is divided into two sub tabs: Geometry and Compute.

**Geometry Tab**
This geometry tab consist of geometry component requires for the heat transfer physics.

**Conductor**
The user interface of conductor required for the heat transfer physics is designed as follows in the figure 3.38;

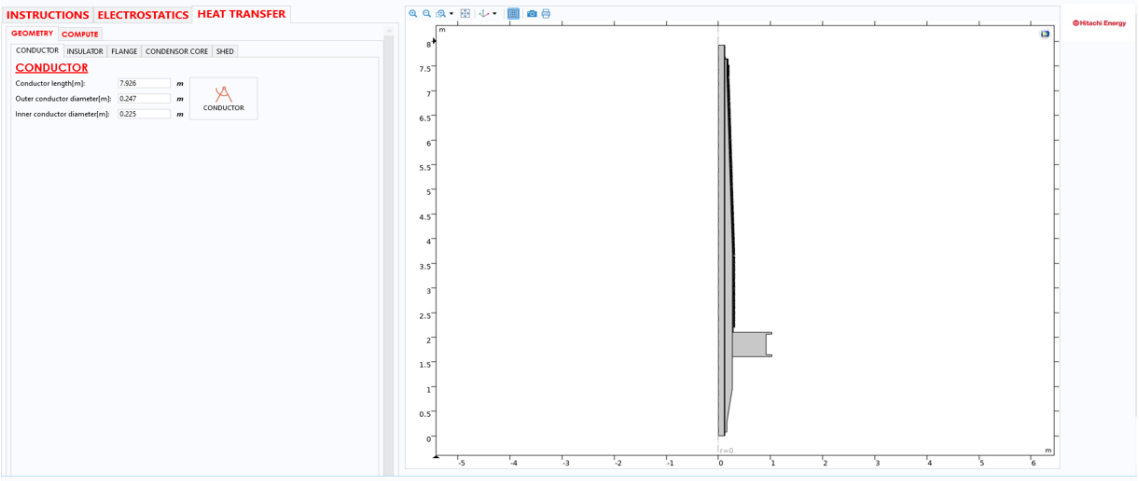



**Figure 3.38:** User interface for the conductor

Similar to the conductor geometry in the electrostatic tab, the length of the conductor, as well as the outer and inner diameters can be adjusted using the input fields provided. These input fields allows for precise customization of the conductor's dimensions to meet the specific simulation requirements. Once the desired dimensions are set the conductor button in the interface can be used to apply these changes. The updated conductor geometry will be displayed in the graphic window allowing the visually verify the modifications.

**Condenser core**
The user interface of condenser core required for the heat transfer physics is designed as follows in the figure 3.39;

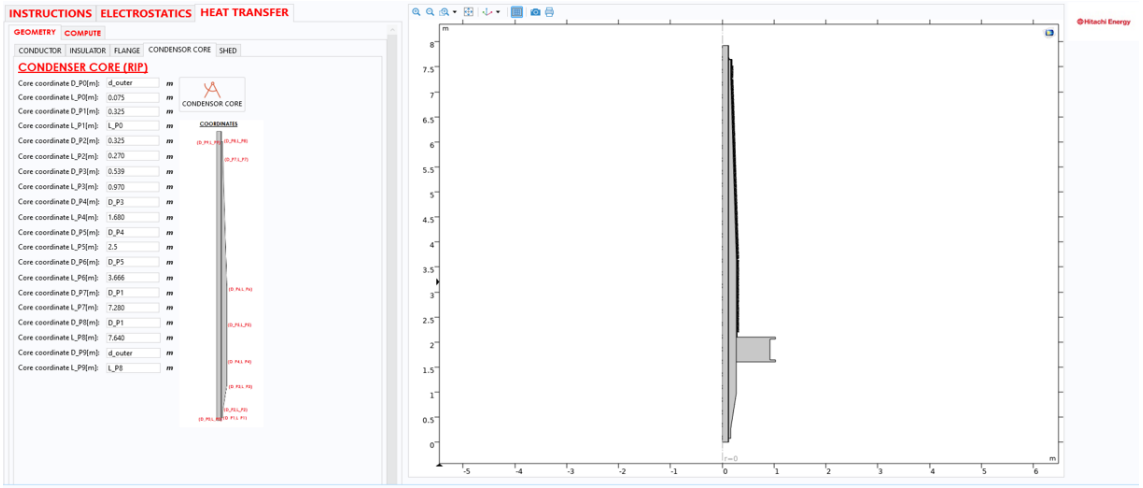



**Figure 3.39:** User interface for the condenser core

Similar to the condenser core geometry in the electrostatic tab, the core coordinates can be adjusted according to the coordinates diagram provided in the interface.These coordinates define the position and orientation of the condenser core within the simulation environment. Using the condenser core button in the interface the changes in the geometry can be verified. When the button is pressed ,the updated geometry is displayed in the graphic window, allows visual confirmation that the modification have been implemented properly.

### Flange

The user interface of the flange is designed as follows in the figure 3.40;

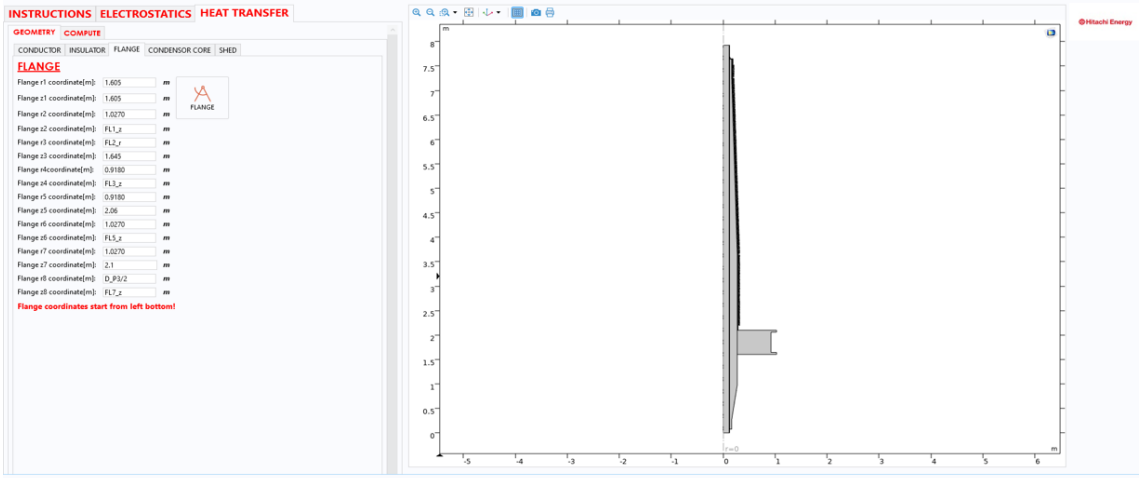



**Figure 3.40:** User interface for the flange

The flange coordinates for heat transfer physics are designed similarly to those in the electrostatics tab, allowing to vary the coordinates using the input fields provided. These coordinates determine the starting point in the layout of the flange, which begins from the bottom left near to the condenser core. By using the flange button in the interface changes can be applied. When the button is pressed the updated

flange geometry is plotted in the graphics window, helps in visually verifying the modification.

### Insulator and Top piece

The user interface for the the Insulator and Top piece is designed as follows in the figure 3.41;

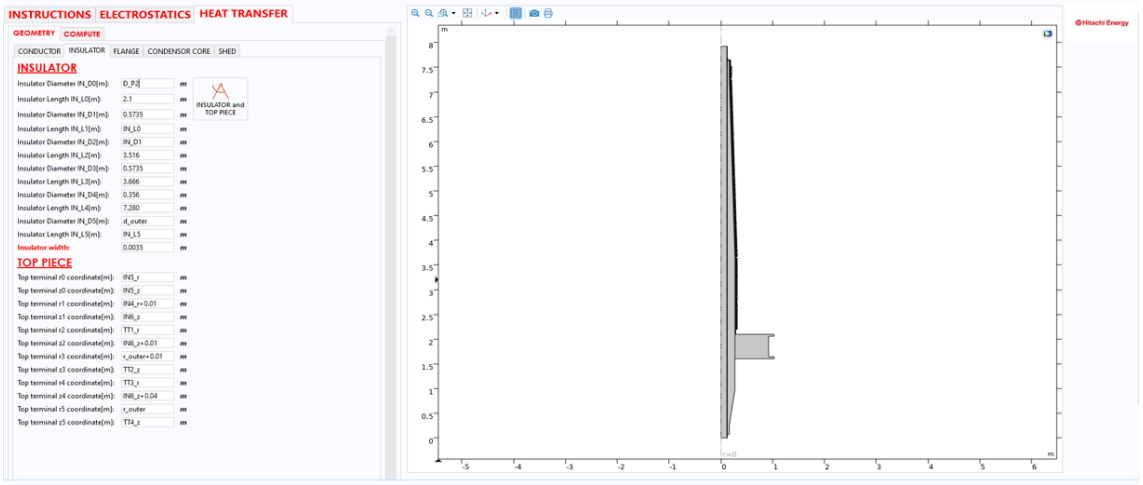



**Figure 3.41:** User Interface for the Insulator and Top piece

The Insulator's diameter and length can be adjusted using the input fields provided in the interface and the insulator width can be modified as needed. Additionally the top piece coordinates allow for precise change to the top piece geometry. To apply and visualize these modification the insulator top piece button can be used. When the button is pressed, the corresponding changes in both the insulator and the top piece are plotted in the graphic window.

### Shed

The user interface for the Sheds are designed as follows in the figure 3.42;

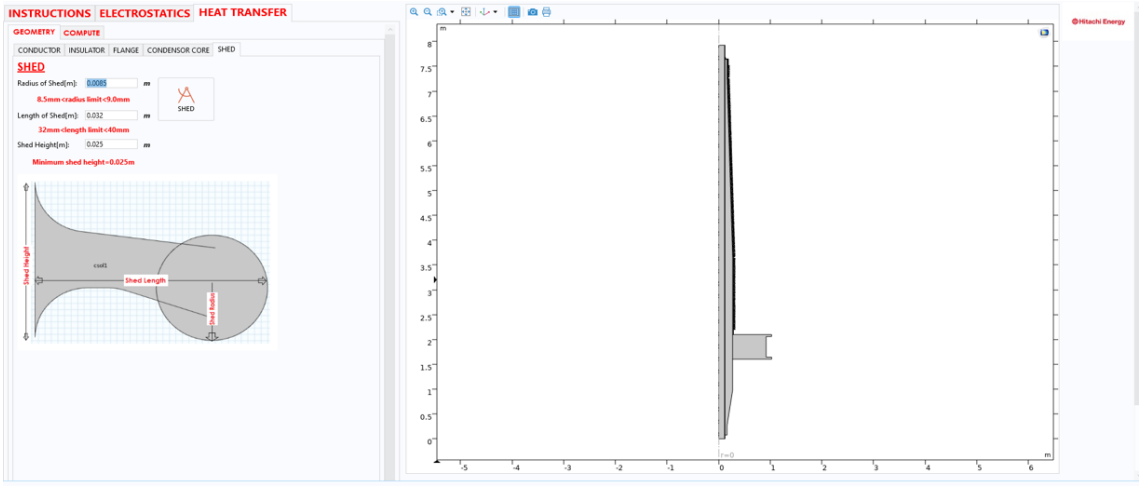



**Figure 3.42:** User interface for Sheds

In the context of heat transfer physics, the user interface focuses solely on the shed geometry, omitting the test tank and corona components. The interface includes input fields and a button dedicated to verifying changes in the shed geometry. The shed's radius, length, and height can be adjusted using the provided input fields. Once the desired values are entered. By clicking on the shed button the shed geometry gets updated and corresponding modification can be displayed in the graphic window.

**Compute Tab**

This tab is dedicated for meshing and computing the geometry for heat transfer physics. The tab look like as follows in the figure 3.43;

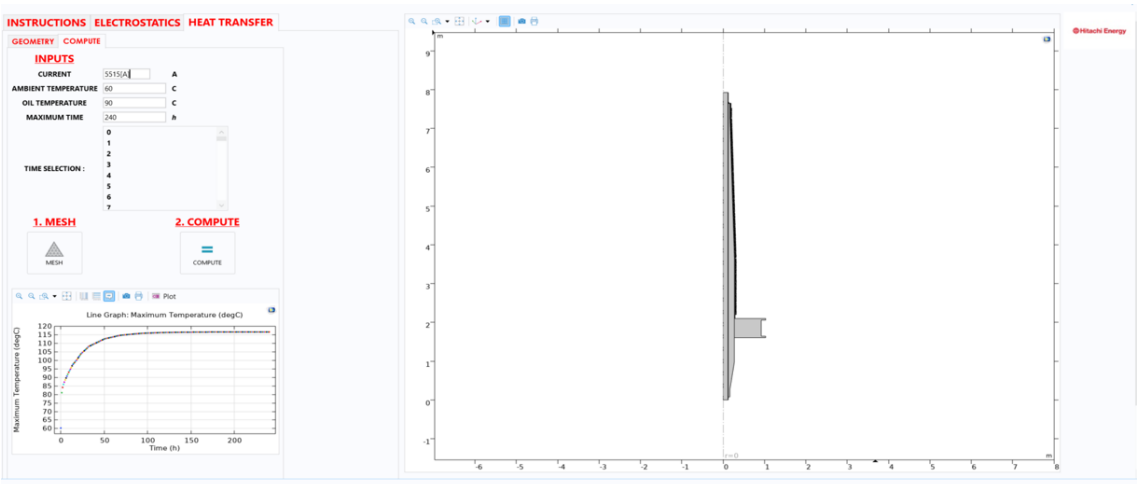



**Figure 3.43:** User interface of Compute tab

The Compute tab is designed to manage the inputs and execution of heat transfer physics simulations. It includes input fields for key parameters such as current, ambient temperature, oil temperature, and maximum simulation time. Current specifies the electrical current applied to the system. Ambient temperature sets the surrounding environmental temperature. Oil temperature defines the temperature of the cooling oil. Maximum time determines the total duration for the simulation runs.

A mesh button is provided to generate a mesh for the entire geometry, ensuring that the model is properly defining the physics controlled mesh. Once the mesh is created, the compute button can be used to run the simulation which perform a time dependent analysis of the heat transfer physics.

Since the primary goal is to analyze the temperature rise, the interface includes a graphical window that plots the maximum temperature over time. This allows user to visualize how the temperature evolves during the simulation. Additionally, a time selection display is available, enabling users to select specific time instants for detailed plotting of the temperature profile along the conductor.

# 4

# Results

## 4.1 Electrostatics

The results for the electrostatics physics after the simulation are presented in the following figures. These figures include data for scenarios both with foils and without foils.

### 4.1.1 With Foils

The following figure 4.1 shows the maximum electric field with foil;

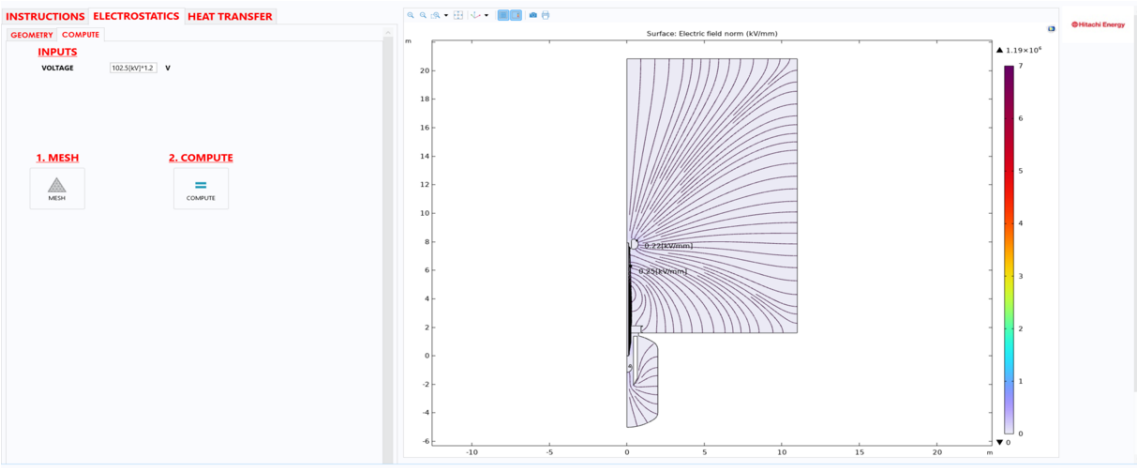



**Figure 4.1:** Maximum Electric field strength with Foils

The simulation shows the maximum electric field strength over the corona is 0.22 kV/mm, while the electric field strength over the shed tips reaches 0.25 kV/mm. This is very close to the values observed in the service simulation. When comparing the service simulation to app simulation,there is a 13% difference in the electric field strength over the corona and a 17% difference in the electric field strength over the shed tips. These discrepancies are primarily due to the simplified geometry used in app simulation, which does not fully capture the complexities of the actual service simulation environment. The geometry in the app simulation was intentionally simplified to streamline the analysis, but this simplification led to slight deviation from the service simulation results.

### 4.1.2 With Only Tap Foil

The following figure 4.2 shows the maximum electric field strength with only tap foil;

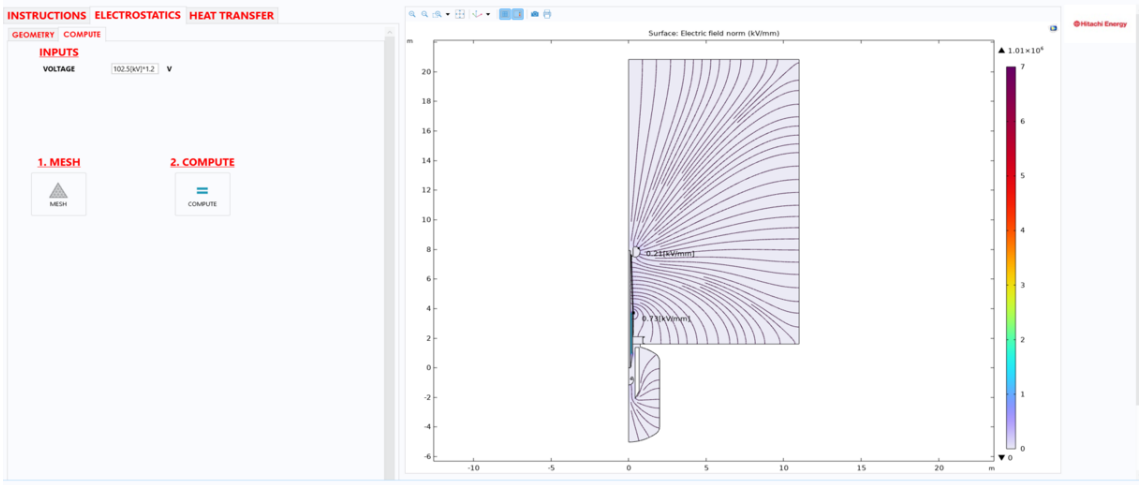



**Figure 4.2:** Maximum Electric field strength with only Tap Foil

The simulation indicates that the maximum electric field strength over the corona is 0.23 kV/mm. Additionally, the electric field strength near the shed is observed to be higher, especially close to the flange, where it reaches 0.75 kV/mm.This increased field strength neat the flange is attributed to the tap foil boundary condition being set to ground. This grounding effect influences the distribution of the electric field, resulting in higher localized value near the flange.

## 4.2 Heat Transfer

The result of the temperature rise test, obtained after running the heat transfer physics simulation are illustrated in the following figure. This detailed figure provides a comprehensive overview of how the temperature evolves within the system during the test. By analysing the temperature distribution over time any potential hotspots of area of concern can be identified. The following figure 4.3 show the result of temperature profile along the conductor at 56 hour.

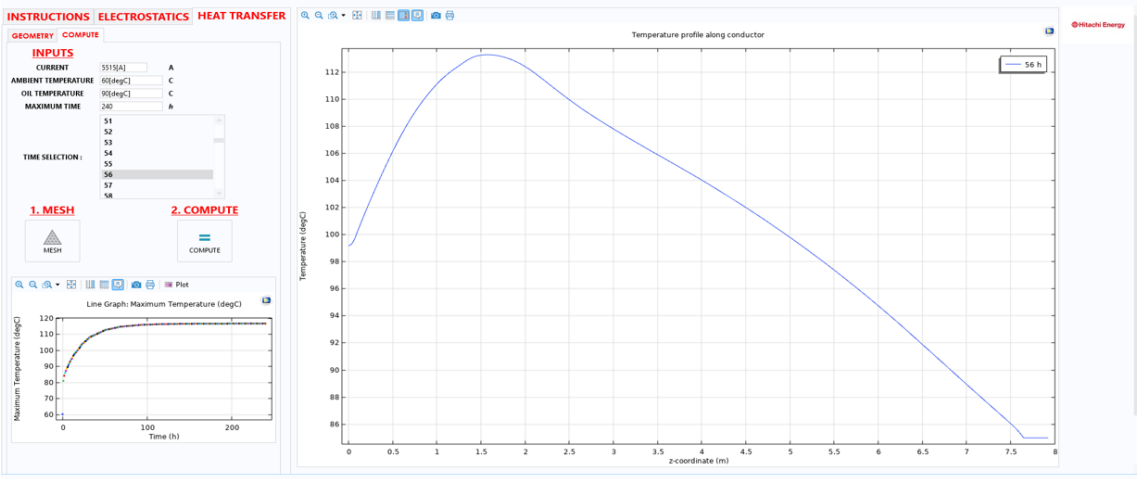



**Figure 4.3:** Temperature profile along conductor

The maximum temperature versus time graph plots the peak temperature corresponding to change in geometry. From this graph, the temperature change $\Delta T$ is less than 1 Kelvin over a 4 hour period can be observed. This specific time at which this condition is met is then identified and selected in the time selection display. Once the appropriate time is selected, the temperature profile along the conductor is plotted for that specific time instant. The resulting temperature profile obtained from both the detailed thermal simulation and the app based thermal simulation are found to be remarkably similar. This alignment between the two sets of results confirms the reliability and accuracy of the app based simulation in replicating the thermal behaviour observed in the more detailed thermal simulation. This validation support the use of the app simulation for practical thermal analysis in engineering application.

# 5
# Discussion & Conclusion

## 5.1 Discussion

The discussion encompasses a comparative analysis between the service simulation and the application simulation. Due to company policies, specific values cannot be disclosed.However, the comparison highlights the difference and similarities in simulation aspect, providing insights into the effectiveness and accuracy of each simulation approach. This analysis is crucial for understanding the potential benefits and limitation of the application simulation in a real-world service context.

**Table 5.1:** Comparison between App simulation and Service simulation

| **Physics** | **Change in app simulation with respect to service simulation** |
|---|---|
| Electrostatics | There was 13% change in electric field (kV/mm) around the corona and 17% change in electric field(kV/mm) at the shed tips. |
| Heat Transfer | Temperature profile along the conductor is mostly similar. |

The comparison table 5.1 indicates that the observed changes in the electric field are attributed to the simplified geometry used in the electrostatic simulations. The exact geometry was not considered, and the complex model was simplified to facilitate easier analysis of the electrostatics. This simplification led to variations in the results between the service simulation and the application simulation.

In contrast, for the heat transfer simulations, the geometry did not play a significant role. As a result, the outcomes of the heat transfer analysis were consistent between both the service simulation and the application simulation, as expected. This consistency underscores the reliability of the heat transfer result, despite the geometric simplifications.

The distinction between the impacts of geometry on electrostatic versus heat transfer physics highlights the importance of model fidelity in different simulation contexts. Understanding these nuances is essential for improving simulation accuracy and optimizing design processes.

## 5.2 Future Recommendation

The application presents several opportunities for future enhancements;

**1.Expanded simulation Capabilities**
Incorporating additional simulations such as DC simulations, thermal service simulations, and the coupling of both AC and DC service simulation would broaden the application's utility and precision.

**2.Corona Optimization**
Further development could include the ability to optimize corona effects by altering the shape and size of components, improving overall performance.

**3.Modular Geometry**
The application could be enhanced to handle different parts of the geometry separately, allowing for more flexibility and detailed customization. This modular approach would also facilitate easier updates and improvements.

**4.Pre-Defined Geometry Libraries**
Creating a directory for text files containing various bushing geometries would streamline the simulation setup process. Engineers could quickly select and implement pre-defined geometries, expediting the design workflow.

## 5.3 Conclusion

The development of this application significantly reduces the time required for designing and simulating various bushing models. The user-friendly interface aids engineers in comprehensively understanding and simulating the models.Consequently, the application not only minimize the manpower needed but also streamlines the design process to considerable extent.
These future improvements recommended would not only make the application more versatile and powerfully but also provide engineers with a more comprehensive tool set for bushing model design and simulation. This ongoing development will continue to enhance efficiency, reduce the need for extensive manual labor, and simplify complex design process.

# Bibliography


[1] Quintero Suárez, F. (2022). Early Anomaly Detection in Electrical Bushings Manufacturing at Hitachi Energy.

[2] COMSOL Mutiphysics,https://www.comsol.com/comsol-multiphysics

[3] B. Tian et al., "Simulation Research on Temperature Rise Characteristics of 252kV Transformer Bushings," 2022 IEEE International Conference on High Voltage Engineering and Applications (ICHVE), Chongqing, China, 2022, pp. 1-4, doi: 10.1109/ICHVE53725.2022.9961643. keywords: Employee welfare;Temperature distribution;Analytical models;Oils;Oil insulation;Flanges;Insulators;bushing;temperature rise;finite element simulation,

[4] J. Liu, D. Cui, B. Li, S. Zhao, J. Jiang and C. Zhang, "Calculation of Transformer Bushing Surface Temperature Considering Dielectric Loss and Solar Radiation," 2023 International Conference on Power Energy Systems and Applications (ICoPESA), Nanjing, China, 2023, pp. 755-760, doi: 10.1109/ICoPESA56898.2023.10141199. keywords: Temperature distribution;Porcelain;Capacitors;Dielectric losses;Oil insulation;Insulators;Transformers;oil-immersed capacitor bushing;temperature field;finite element simulation;thermal radiation;dielectric loss,

[5] COMSOL Mutiphysics,https://www.comsol.com/multiphysics/electrostatics-theory

[6] COMSOL Multiphysics,*COMSOL Application Builder Manual*,Version 5.3,2017.Available at:https://doc.comsol.com/5.3/doc/com.comsol.help.comsol/COMSOL_ApplicationBuilderManual.pdf

# A

# Appendix

The following Java code was orginally written by Maria Arvidsson, a new employee at Hitachi Energy. It has been revised and enhanced to meet the application's specification, with additional functionalities incorporated as required.

## A.1 Creating Foils in Geometry

```
// Reading data
String[][] excelFoil = readExcelFile("
    condenser_core_parameters_GSC1450.xlsx", "ANALYSIS", "D16"
    ); // diameter and length of foils
String[][] excelPorF = readExcelFile("
    condenser_core_parameters_GSC1450.xlsx", "ANALYSIS", "C16"
    ); // full or partial foil
String[][] excelOD = readExcelFile("
    condenser_core_parameters_GSC1450.xlsx", "DESIGN", "F13");

int rows = excelFoil.length; //Number of rows in excel sheet
    "ANALYSIS", equals number of foils
int columns = 5;
double[][] excelData = new double[rows][columns];

// Turning string matrix into matrix with numbers using for
    loops. For tube/foil
for (int i = 0; i < rows; i++) { //looping over every row
  for (int j = 0; j < columns; j++) {
    double value = Double.parseDouble(excelFoil[i][j]); //
       Changing type from String to Double.
    double scale = 0.001; //Excel sheet value is in [mm].
       Scaling to [m] for COMSOL
    excelData[i][j] = value*scale; //Changing [mm] to [m]
  }
}

// Data for if the foil is partial or full (PorF)
int row = excelPorF.length;

for (int k = 0; k < row; k++) {
```

```
if (excelData[k][2] == excelData[k][3]) { // Create full
    foils, if the cells are the same then it is a full foil
  String foilTag = "bF"+k+" full"; //Name of foil
  model.component("comp1").geom("geom1").create(foilTag, "
     BezierPolygon"); //Creating geometry object
  model.component("comp1").geom("geom1").feature(foilTag).
     set("type", "solid"); //Setting geometry object
     features
  model.component("comp1").geom("geom1").feature(foilTag).
     set("p", new double[][]{{excelData[k][0]/2, excelData[
     k][0]/2}, {excelData[k][1], excelData[k][4]}}); //foil
      placement
  model.component("comp1").geom("geom1").feature(foilTag).
     set("degree", 1); //1D object
  model.component("comp1").geom("geom1").run(foilTag); //
     Building foil

  // Adding a cumulative selection for each full foil
  String nameFF = "FF"+k; // This is the label for the
     cumulative selection, FF = Full Foil
  String selFF = "selFF"+k; // This is the "name" of the
     cumulative selection
  model.component("comp1").geom("geom1").feature(foilTag).
     set("w", new int[]{1, 1});
  model.component("comp1").geom("geom1").selection().create
     (selFF, "CumulativeSelection");
  model.component("comp1").geom("geom1").selection(selFF).
     label(nameFF);
  model.component("comp1").geom("geom1").feature(foilTag).
     set("contributeto", selFF);
} else { //Creating upper partial foil
  String foilTag1 = "b"+k+" partial upp"; //Name of upper
     partial foil
  model.component("comp1").geom("geom1").create(foilTag1, "
     BezierPolygon"); //Creating geometry object
  model.component("comp1").geom("geom1").feature(foilTag1).
     set("type", "solid"); //Setting geometry object
     features
  model.component("comp1").geom("geom1").feature(foilTag1).
     set("p", new double[][]{{excelData[k][0]/2, excelData[
     k][0]/2}, {excelData[k][1], excelData[k][2]}}); //foil
      placement
  model.component("comp1").geom("geom1").feature(foilTag1).
     set("degree", 1); //1D object
  model.component("comp1").geom("geom1").run(foilTag1); //
     Building foil

  // Adding a cumulative selection for each upper partial
     foil
```

```
    String namePFU = "PFU"+k; // PFU = Partial Foil Upper
    String selPFU = "selPFU"+k;
    model.component("comp1").geom("geom1").feature(foilTag1).
       set("w", new int[]{1, 1});
    model.component("comp1").geom("geom1").selection().create
       (selPFU, "CumulativeSelection");
    model.component("comp1").geom("geom1").selection(selPFU).
       label(namePFU);
    model.component("comp1").geom("geom1").feature(foilTag1).
       set("contributeto", selPFU);

    // Creating lower partial foil
    String foilTag2 = "b"+k+" partial low"; //Name of lower
       partial foil
    model.component("comp1").geom("geom1").create(foilTag2, "
       BezierPolygon"); //Creating geometry object
    model.component("comp1").geom("geom1").feature(foilTag2).
       set("type", "solid"); //Setting geometry object
       features
    model.component("comp1").geom("geom1").feature(foilTag2).
       set("p", new double[][]{{excelData[k][0]/2, excelData[
       k][0]/2}, {excelData[k][3], excelData[k][4]}}); //foil
        placement
    model.component("comp1").geom("geom1").feature(foilTag2).
       set("degree", 1); // 1D object
    model.component("comp1").geom("geom1").run(foilTag2); //
       Building foil

    // Adding a cumulative selection for each lower partial
       foil
    String namePFL = "PFL"+k; // PFL = Partial Foil Lower
    String selPFL = "selPFL"+k;
    model.component("comp1").geom("geom1").feature(foilTag2).
       set("w", new int[]{1, 1});
    model.component("comp1").geom("geom1").selection().create
       (selPFL, "CumulativeSelection");
    model.component("comp1").geom("geom1").selection(selPFL).
       label(namePFL);
    model.component("comp1").geom("geom1").feature(foilTag2).
       set("contributeto", selPFL);
  }
}
//
   ------------------------------------------------------------------------

// Enclose the foils in order to have a closed geometry
// Connect the end points of the foils and connect with a
   line to the conductor.
```

```
// 1. Draw lines between upper foils
for (int m = 0; m < rows-1; m++) {
  String UpperLine = "L"+m+" upper line"; //Name of upper
     partial foil
  model.component("comp1").geom("geom1").create(UpperLine, "
     LineSegment"); //Creating geometry object
  model.component("comp1").geom("geom1").feature(UpperLine).
     set("specify1", "coord"); //Setting geometry object
     features
  model.component("comp1").geom("geom1").feature(UpperLine).
     set("specify2", "coord");
  model.component("comp1").geom("geom1").feature(UpperLine).
     set("coord1", new double[]{excelData[m][0]/2, excelData[
     m][1]}); // Start point
  model.component("comp1").geom("geom1").feature(UpperLine).
     set("coord2", new double[]{excelData[m+1][0]/2,
     excelData[m+1][1]}); // End point
  model.component("comp1").geom("geom1").run(UpperLine); //
     Creating line
}

// 2. Draw lines between lower foils
for (int m = 0; m < rows-1; m++) {
  String LowerLine = "L"+m+" lower line"; //Name of upper
     foil
  model.component("comp1").geom("geom1").create(LowerLine, "
     LineSegment"); //Creating geometry object
  model.component("comp1").geom("geom1").feature(LowerLine).
     set("specify1", "coord"); //Setting geometry object
     features
  model.component("comp1").geom("geom1").feature(LowerLine).
     set("specify2", "coord");
  model.component("comp1").geom("geom1").feature(LowerLine).
     set("coord1", new double[]{excelData[m][0]/2, excelData[
     m][4]}); // Start point
  model.component("comp1").geom("geom1").feature(LowerLine).
     set("coord2", new double[]{excelData[m+1][0]/2,
     excelData[m+1][4]}); // End point
  model.component("comp1").geom("geom1").run(LowerLine); //
     Creating line
}

//
   ------------------------------------------------------------------

// 3. Draw lines between the conductor and the foils

// For outer diameter (OD)
int rowsOD = 1;
```

```
int columnsOD = excelOD[0].length;
double[][] OD = new double[rowsOD][columnsOD];

for (int c = 0; c < rowsOD; c++) {
  for (int d = 0; d < columnsOD; d++) {
    if (excelOD[c][d] == "String[][]") {
      excelOD[c][d] = "0";
    } else {
      double valueOD = Double.parseDouble(excelOD[c][d]);
      double scaleOD = 0.001;
      OD[c][d] = valueOD*scaleOD; // converting [mm] to [m]
    }
  }
}

// maybe can be removed? Long line along the conductor
String Line = "L"; //Name of upper partial foil
model.component("comp1").geom("geom1").create(Line, "
   LineSegment"); //Creating geometry object
model.component("comp1").geom("geom1").feature(Line).set("
   specify1", "coord"); //Setting geometry object features
model.component("comp1").geom("geom1").feature(Line).set("
   specify2", "coord");
model.component("comp1").geom("geom1").feature(Line).set("
   coord1", new double[]{OD[0][0]/2, excelData[0][1]}); //
   Start point, r motsvarar x-axeln och z  r  y-axeln
model.component("comp1").geom("geom1").feature(Line).set("
   coord2", new double[]{OD[0][0]/2, excelData[0][4]}); //
   End point
model.component("comp1").geom("geom1").run(Line); // Creating
    line

// Line between the upper foil and the conductor
String LineU = "LU";
model.component("comp1").geom("geom1").create(LineU, "
   LineSegment"); //Creating geometry object
model.component("comp1").geom("geom1").feature(LineU).set("
   specify1", "coord"); //Setting geometry object features
model.component("comp1").geom("geom1").feature(LineU).set("
   specify2", "coord");
model.component("comp1").geom("geom1").feature(LineU).set("
   coord1", new double[]{OD[0][0]/2, excelData[0][1]}); //
   Start point, r motsvarar x-axeln och z  r  y-axeln
model.component("comp1").geom("geom1").feature(LineU).set("
   coord2", new double[]{excelData[0][0]/2, excelData[0][1]})
   ; // End point
model.component("comp1").geom("geom1").run(LineU); //
   Creating line
```

```
// Line between the lower foil and the conductor
String LineL = "LL";
model.component("comp1").geom("geom1").create(LineL, "
    LineSegment"); //Creating geometry object
model.component("comp1").geom("geom1").feature(LineL).set("
    specify1", "coord"); //Setting geometry object features
model.component("comp1").geom("geom1").feature(LineL).set("
    specify2", "coord");
model.component("comp1").geom("geom1").feature(LineL).set("
    coord1", new double[]{OD[0][0]/2, excelData[0][4]}); //
    Start point, r motsvarar x-axeln och z  r  y-axeln
model.component("comp1").geom("geom1").feature(LineL).set("
    coord2", new double[]{excelData[0][0]/2, excelData[0][4]})
    ; // End point
model.component("comp1").geom("geom1").run(LineL); //
    Creating line

//
    ------------------------------------------------------------------

// 4. Create line between partial foils on the inner side

// Drawing lines over full foils on the upper side
for (int k = 0; k < rows-2; k++) {
  if (excelData[k][2] == excelData[k][3] && k > 2 &&
      excelData[k+1][2] != excelData[k+1][3]) { // Check if it
       is a full foil
     String UpperLine5 = k+"doubleU"; // A line between two
        partial foils if there is a full foil in between
     model.component("comp1").geom("geom1").create(UpperLine5,
         "LineSegment"); //Creating geometry object
     model.component("comp1").geom("geom1").feature(UpperLine5
        ).set("specify1", "coord"); //Setting geometry object
        features
     model.component("comp1").geom("geom1").feature(UpperLine5
        ).set("specify2", "coord");
     model.component("comp1").geom("geom1").feature(UpperLine5
        ).set("coord1", new double[]{excelData[k-1][0]/2,
        excelData[k-1][2]}); // Start point
     model.component("comp1").geom("geom1").feature(UpperLine5
        ).set("coord2", new double[]{excelData[k+1][0]/2,
        excelData[k+1][2]}); // End point
     model.component("comp1").geom("geom1").run(UpperLine5);
  }
}
// Drawing lines between partial foils on the upper side
for (int k = 0; k < rows-1; k++) {
  if (excelData[k][2] != excelData[k][3] && excelData[k+1][2]
      != excelData[k+1][3]) { // Check if it is a partial
```

```
      foil and that the next one is a partial foil
    String UpperLine0 = k+"simpleU"; // A line between two
       partial foils
    model.component("comp1").geom("geom1").create(UpperLine0,
        "LineSegment"); //Creating geometry object
    model.component("comp1").geom("geom1").feature(UpperLine0
       ).set("specify1", "coord"); //Setting geometry object
       features
    model.component("comp1").geom("geom1").feature(UpperLine0
       ).set("specify2", "coord");
    model.component("comp1").geom("geom1").feature(UpperLine0
       ).set("coord1", new double[]{excelData[k][0]/2,
       excelData[k][2]}); // Start point
    model.component("comp1").geom("geom1").feature(UpperLine0
       ).set("coord2", new double[]{excelData[k+1][0]/2,
       excelData[k+1][2]}); // End point
    model.component("comp1").geom("geom1").run(UpperLine0);
  }
}

//
   ------------------------------------------------------------------

// Drawing lines over full foils on the lower side
for (int k = 0; k < rows-2; k++) {
  if (excelData[k][2] == excelData[k][3] && k > 2 &&
     excelData[k+1][2] != excelData[k+1][3]) { // Check if it
      is a full foil
    String UpperLine5 = k+"doubleL"; // A line between two
       partial foils if there is a full foil in between
    model.component("comp1").geom("geom1").create(UpperLine5,
        "LineSegment"); //Creating geometry object
    model.component("comp1").geom("geom1").feature(UpperLine5
       ).set("specify1", "coord"); //Setting geometry object
       features
    model.component("comp1").geom("geom1").feature(UpperLine5
       ).set("specify2", "coord");
    model.component("comp1").geom("geom1").feature(UpperLine5
       ).set("coord1", new double[]{excelData[k-1][0]/2,
       excelData[k-1][3]}); // Start point
    model.component("comp1").geom("geom1").feature(UpperLine5
       ).set("coord2", new double[]{excelData[k+1][0]/2,
       excelData[k+1][3]}); // End point
    model.component("comp1").geom("geom1").run(UpperLine5);
  }
}
// Drawing lines between partial foils on the lower side
for (int k = 0; k < rows-1; k++) {
```

```
  if (excelData[k][2] != excelData[k][3] && excelData[k+1][2]
      != excelData[k+1][3]) { // Check if it is a partial
     foil and that the next one is a partial foil
    String UpperLine0 = k+"simpleL"; // A line between two
       partial foils
    model.component("comp1").geom("geom1").create(UpperLine0,
        "LineSegment"); //Creating geometry object
    model.component("comp1").geom("geom1").feature(UpperLine0
       ).set("specify1", "coord"); //Setting geometry object
       features
    model.component("comp1").geom("geom1").feature(UpperLine0
       ).set("specify2", "coord");
    model.component("comp1").geom("geom1").feature(UpperLine0
       ).set("coord1", new double[]{excelData[k][0]/2,
       excelData[k][3]}); // Start point
    model.component("comp1").geom("geom1").feature(UpperLine0
       ).set("coord2", new double[]{excelData[k+1][0]/2,
       excelData[k+1][3]}); // End point
    model.component("comp1").geom("geom1").run(UpperLine0);
  }
}
model.component("comp1").geom("geom1").feature("pol2").active
   (false);
useGraphics(model.component("comp1").geom("geom1"), "main/
   graphics1");
```

## A.2 Adding Floating Potential to Foils

```
// Code for adding potentials to the foils

// Read data from Excel
String[][] excelFoil = readExcelFile("
   condenser_core_parameters_GSC1450.xlsx", "ANALYSIS", "D16"
   ); // diameter and length of foils

int rows = excelFoil.length; //Number of rows in excel sheet
   "ANALYSIS", equals number of foils
int columns = 5;
double[][] excelData = new double[rows][columns];

// Turning string matrix into matrix with numbers using for
   loops. For tube/foil
for (int i = 0; i < rows; i++) { //looping over every row
  for (int j = 0; j < columns; j++) {
    double value = Double.parseDouble(excelFoil[i][j]); //
       Changing type from String to Double.
```

```
    double scale = 0.001; //Excel sheet value is in [mm].
       Scaling to [m] for COMSOL
    excelData[i][j] = value*scale; //Changing [mm] to [m]
  }
}

// Creating the physics, in this case "es", add "ec" as well?
model.component("comp1").physics("es").active(true);

// Adding a potential to each foil depending on if it is a
   full, upper partial or lower partial foil
for (int k = 1; k < rows; k++) {
  if (excelData[k][2] == excelData[k][3]) { // Add potential
     to a full foil
    //String potFull = "potFF"+k; // Object for the electric
       potential
    //String nameFF = "geom1_selFF"+k+"_bnd"; // Name for the
       electric potential
    String potF = "potF"+k; // Object for the floating
       potential
    String nameF = "geom1_selFF"+k+"_bnd"; // Name for the
       floating potential
    //model.component("comp1").physics("es").create(potFull,
       "ElectricPotential", 1); // Adding electric potential
    //model.component("comp1").physics("es").feature(potFull)
       .selection().named(nameFF);
    if (k == rows-1) {
      String gndF = "gndF"+k; //object for tap foil
      String nameG = "geom1_selFF"+k+"_bnd"; //name for tap
         foil
      model.component("comp1").physics("es").create(gndF, "
         Ground", 1); //adding ground for tap foil
      model.component("comp1").physics("es").feature(gndF).
         selection().named(nameG);
    }
    else {
      model.component("comp1").physics("es").create(potF, "
         FloatingPotential", 1); // Adding floating potential
      model.component("comp1").physics("es").feature(potF).
         selection().named(nameF);
    }
  }

  else {
    // Add potential to the upper partial foils
    String potU = "potUPF"+k; // UPF = Upper Partial Foil
    String nameU = "geom1_selPFU"+k+"_bnd"; // Name of the
       potential for the upper partial foil
```

```
    model.component("comp1").physics("es").create(potU, "
       FloatingPotential", 1);
    model.component("comp1").physics("es").feature(potU).
       selection().named(nameU);

    // Add potential to the upper partial foils
    String potL = "potLPF"+k; // LPF = Lower Partial Foil
    String nameL = "geom1_selPFL"+k+"_bnd";
    model.component("comp1").physics("es").create(potL, "
       FloatingPotential", 1);
    model.component("comp1").physics("es").feature(potL).
       selection().named(nameL);
  }
}
```

## A.3 Deleting Foils

```
//Method that deletes the foils.
// Get data from Excel
String[][] excelFil = readExcelFile("
   condenser_core_parameters_GSC1450.xlsx", "ANALYSIS", "D16"
   ); // Needs to be same input as "CreateFoils" method.


// Remove the single lines to the conductor
model.component("comp1").geom("geom1").feature().remove("L");
    //Remove the line on the outer rectangle
model.component("comp1").geom("geom1").feature().remove("LU")
   ; //Remove the upper line between the outer rectangle and
   the foil
model.component("comp1").geom("geom1").feature().remove("LL")
   ; //Remove the lower line between the outer rectangle and
   the foil

// Removing every foil
int rows = excelFil.length;
for (int i = 0; i < rows; i++) { // looping over every row (#
   rows = #foils)
  try { // Remove full foil object
    model.component("comp1").geom("geom1").feature().remove("
       bF"+i+" full"); //Removing the full foil
  }
  catch (Exception e1) { // If no full foil object ==> two
     partial foils are removed instead.
    model.component("comp1").geom("geom1").feature().remove("
       b"+i+" partial upp"); //Removing partial foil
```

```
    model.component("comp1").geom("geom1").feature().remove("
       b"+i+" partial low"); //Removing partial foil
  }
}

//
   ----------------------------------------------------------------

// Removing every upper line between foils
for (int j = 0; j < rows-1; j++) {
  model.component("comp1").geom("geom1").feature().remove("L"
     +j+" upper line");
  model.component("comp1").geom("geom1").feature().remove("L"
     +j+" lower line");
}

// Removing lines on the inside of the parial foils
int columns = 5;
double[][] excelData = new double[rows][columns]; // message(
   excelData);

// Turning string matrix into matrix with numbers using for
   loops. For tube/foil
for (int i = 0; i < rows; i++) { //looping over every row
  for (int j = 0; j < columns; j++) {
    double value = Double.parseDouble(excelFil[i][j]); //
       Changing type from String to Double.
    double scale = 0.001; //Excel sheet value is in [mm].
       Scaling to [m] for COMSOL
    excelData[i][j] = value*scale; //Changing [mm] to [m]
  }
}

//
   ----------------------------------------------------------------

// Removing every line between foils on the inside of the
   partial foils
for (int k = 0; k < rows-2; k++) {
  if (excelData[k][2] == excelData[k][3] && k > 2 &&
     excelData[k+1][2] != excelData[k+1][3]) { // Checking if
      there is a full foil between the partial foils
    model.component("comp1").geom("geom1").feature().remove(k
       +"doubleU"); // Remove the line overlapping a full
       foil on the upper side
    model.component("comp1").geom("geom1").feature().remove(k
       +"doubleL"); // Remove the line overlapping a full
       foil on the lower side
  }
```

```
}

for (int k = 0; k < rows-1; k++) {
  if (excelData[k][2] != excelData[k][3] && excelData[k+1][2]
      != excelData[k+1][3]) { // Checking if it is a partial
     foil and the next one in a partial foil
    model.component("comp1").geom("geom1").feature().remove(k
       +"simpleU"); // Remove line between partial foils on
       the upper side
    model.component("comp1").geom("geom1").feature().remove(k
       +"simpleL"); // Remove line between partial foils on
       the lower side
  }
}
```

## A.4 Deleting Cummilative Selection

```
// Delete the cummilative-section
model.component("comp1").geom("geom1").selection().remove("
   cumulativeSelection");

//
   ------------------------------------------------------------------------

// Get data from Excel
String[][] excelFoil = readExcelFile("
   condenser_core_parameters_GSC1450.xlsm", "ANALYSIS", "D16"
   ); // diameter and length of foils

int rows = excelFoil.length; //Number of rows in excel sheet
   "ANALYSIS", equals number of foils
int columns = 5;
double[][] excelData = new double[rows][columns];

// Turning string matrix into matrix with numbers using for
   loops. For tube/foil
for (int i = 0; i < rows; i++) { //looping over every row
  for (int j = 0; j < columns; j++) {
    double value = Double.parseDouble(excelFoil[i][j]); //
       Changing type from String to Double.
    double scale = 0.001; //Excel sheet value is in [mm].
       Scaling to [m] for COMSOL
    excelData[i][j] = value*scale; //Changing [mm] to [m]
  }
}
```

```
// Deleting the cummilativeselection of each foil depending
   on if it is a full, upper partial or lower partial foil
for (int k = 0; k < rows; k++) {
  if (excelData[k][2] == excelData[k][3]) { // Add potential
     to a full foil

    String selFF = "selFF"+k; // deleting the cummilative
       selection for the full foil
    model.component("comp1").geom("geom1").selection().remove
       (selFF);

  } else {

    String selPFU = "selPFU"+k; //deleting the cummilative
       selection for the partial foil upper
    model.component("comp1").geom("geom1").selection().remove
       (selPFU);

    String selPFL = "selPFL"+k; //deldting the cummilative
       selection  for the partial foil lower
    model.component("comp1").geom("geom1").selection().remove
       (selPFL);

  }
}

useGraphics(model.component("comp1").geom("geom1"), "main/
   graphics1");
```

## A.5 Deleting Floating Potential

```
// Delete the electrostatics-section
model.component("comp1").physics("es").feature().remove("es")
   ;

//
   ------------------------------------------------------------------

// Get data from Excel
String[][] excelFoil = readExcelFile("
   condenser_core_parameters_GSC1450.xlsx", "ANALYSIS", "D16"
   ); // diameter and length of foils

int rows = excelFoil.length; //Number of rows in excel sheet
   "ANALYSIS", equals number of foils
int columns = 5;
```

```
double[][] excelData = new double[rows][columns];

// Turning string matrix into matrix with numbers using for
   loops. For tube/foil
for (int i = 0; i < rows; i++) { //looping over every row
  for (int j = 0; j < columns; j++) {
    double value = Double.parseDouble(excelFoil[i][j]); //
       Changing type from String to Double.
    double scale = 0.001; //Excel sheet value is in [mm].
       Scaling to [m] for COMSOL
    excelData[i][j] = value*scale; //Changing [mm] to [m]
  }
}

// Deleting the potential of each foil depending on if it is
   a full, upper partial or lower partial foil
for (int k = 1; k < rows; k++) {
  if (excelData[k][2] == excelData[k][3]) { // Add potential
     to a full foil
    // model.component("comp1").physics("es").feature().
       remove("potFF"+k);
    model.component("comp1").physics("es").feature().remove("
       potF"+k);
    model.component("comp1").physics("es").feature().remove("
       gndF"+k);
  } else {
    // Add potential to the upper partial foils
    // String potU = "potUPF"+k; // UPF = Upper Partial Foil
    // String nameU = "geom1_selPFU"+k+"_bnd"; // Name of the
        potential for the upper partial foil
    model.component("comp1").physics("es").feature().remove("
       potUPF"+k);

    // Add potential to the upper partial foils
    // String potL = "potLPF"+k; // LPF = Lower Partial Foil
    // String nameL = "geom1_selPFL"+k+"_bnd";
    model.component("comp1").physics("es").feature().remove("
       potLPF"+k);
  }
}
```

**DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY**
**CHALMERS UNIVERSITY OF TECHNOLOGY**
Gothenburg, Sweden
www.chalmers.se


CHALMERS
UNIVERSITY OF TECHNOLOGY