Institute of Electrical Engineering and Information Technology
Paderborn University
Department of Power Electronics and Electrical Drives
Prof. Dr.-Ing. Joachim Böcker

Documentation

Extension of the FEM Magnetics Toolbox for n-winding Transformers

by

Othman Abujazar

Student ID: 6920221

Supervisor: Till Piepenbrock

Supervisor: Nikolas Förster

Filing Date: April 24, 2023

Declaration of Authorship

I declare that I have authored this thesis independently, that I have not used other than the declared sources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources. This paper was not previously presented to another examination board and has not been published.

Paderborn, _____ _____

Date Signature

Abstract

This project aims to extend the capabilities of the Finite Element Method Magnetics Toolbox (FEMMT) by adding the ability to model transformers with any number of windings. FEMMT is an open-source toolbox built by the Department of Power Electronics and Electrical Drives (LEA) at Paderborn University for 2D radial symmetrical FEM simulation. The project involves replacing the existing code that handles two-winding transformers with a dynamic "for-loop", allowing the program to iterate through each winding in the transformer and define the necessary variables. Additionally, a new method called "*TenCells_Split*" is developed to divide the winding window into ten virtual windows for ten windings. A single frequency sweep simulation of a ten-winding transformer was performed , showing good agreement. This project enhances the versatility and adaptability of FEMMT, making it a more valuable tool for a wide range of applications in transformer design and simulation.

Contents

List of Figures

List of Tables

Listings

1 Introduction

As the transformer is one of the most important devices in electricity, projects of all kinds may need additional windings in transformers to perform some new tasks, and for this reason the project focuses on developing some of its parts in FEM magnetics toolbox (FEMMT); built by the Department of Power Electronics and Electrical Drives (LEA) at Paderborn University. FEMMT allows drawing a core and winding forms for inductor and different types of transformer with the help of python scripts for (2D radial symmetrical) in FEM simulation, working in an open source environment.

The project aims to extend FEMMT project with the capability of building n-windings and validating the simulation of n-winding transformer, comparing the results with Finite Element Method Magnetics (FEMM 4.2). FEMM 4.2 is a set of programs that can solve electromagnetic problems with low frequency on 2D planer and axisymmetric domains, which is builded by PhD. David Meeker.

In order to achieve this goal, the existing code in FEMMT is modified to handle transformers with any number of windings using dynamic lists and for-loops, making the code more adaptable and versatile. Additionally, a new method is implemented to split the virtual winding window into ten smaller windows for ten-winding transformers.

To validate the simulation results, a single frequency sweep simulation was performed on a three-winding transformer, and the results were compared to the simulation results obtained using FEMM 4.2. The simulation results showed good agreement between the two tools, indicating that the modifications made to FEMMT were successful in allowing it to handle n-winding transformers. a single frequency sweep simulation was performed also on a ten-winding transformer, showing logical results.

Overall, the extension of FEMMT with the capability of building n-windings and validating the simulation of n-winding transformers provides a valuable tool for the design and analysis of transformers with multiple windings.

2 Transformer

2.1 Principle of Transformer

The transformer is an electrical device that can convert ac electrical energy at on voltage stage to ac electrical energy at another voltage stage. It is one of the most ubiquitous devices in modern daily life as it used everywhere. The basis of transformer action depends on time-changing magnetic field, which can induce a voltage in a coil of wire (or winding) if it passes through the coil. The concept of a magnetic field production can be explained via Ampere's law which is:

$$\oint_L H \cdot dl = I_{net} \tag{2.1}$$

where H is the magnetic field intensity in ampere-turns per meter, and dl is a differential element of length along the path of integration as shown in Fig. 2.1.

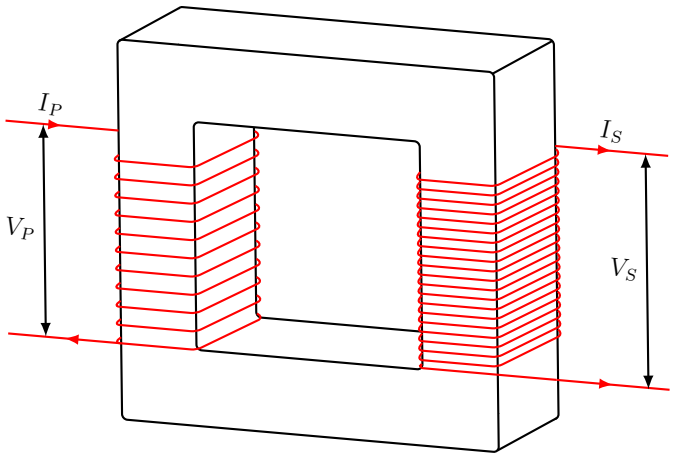

Fig. 2.1: A simple magnetic core

The basic understanding of this equation states that if a current passes through N turns of winding wrapped about one leg of an iron's core , the magnetic field produced by

this current will remain inside the core, which means that the mean path length of the core is the path of integration. As the path of integration is cut by a number of turns of coil, then Ampere's law becomes:

$$Hl = Ni \tag{2.2}$$

meaning that the magnitude of the magnetic field intensity is:

$$H = \frac{Ni}{l} \tag{2.3}$$

With the help of the relationship between the magnetic field intensity and magnetic field density, the magnetic field density can be written as:

$$B = \mu H = \frac{\mu Ni}{l} \tag{2.4}$$

where H = magnetic field intensity (ampere-turns per meter), μ = magnetic permeability, and B = magnetic field density (Tesla). Now the total flux (ϕ) through a closed surface is the integral of the magnetic field density (B), which is:

$$\phi = \oint_A B \cdot dA \tag{2.5}$$

where the unit of fluxes is weber.

Now if it is assumed that that flux density is constant through the area A, and the vector B is perpendicular to the plane, then the generated equation is:

$$\phi = BA = \frac{\mu NiA}{l} \tag{2.6}$$

Equation 2.6 is similar to the logic of ohm's law in electric circuits, which leads to magnetomotive force as in with respect to the electromotive force in magnetic circuits. Magnetic circuits is widely used in lieu of the complexities of Maxwell's equations.

In the other Hand, there is also number of turns of coils wrapped about the other leg of the core. This will leads to another important concept of the transformer, which is Faraday's law. Faraday's law states that a time-changing magnetic field or simply flux passing through a turn of winding will induce a voltage in the turn of winding which is directly proportional to the rate of change with respect to time. This can be written in equation as:

$$e_{ind} = -N\frac{d\phi}{dt} \tag{2.7}$$

where e_{ind} is the voltage induced in the turn of winding and N is number of turns in the second leg. The minus here is due to Lenz's law which is present to foresee the polarity of the voltage induced in the coils. The Ampere's and Faraday's laws are the main concepts to figure out how a transformer generally works. In addition, by understanding these laws, it is possible to talk about the losses of all kinds and its causes.

2.2 Losses in transformer

As the resistance causes losses in electrical circuits, the reluctance can play same issue in magnetic circuits. The reluctance of a magnetic circuit can be described as:

$$\Re = \frac{l}{\mu A} \tag{2.8}$$

where its units are ampere-turns per weber. It is mentioned already that the magnetic circuits have only approximation results due to many reasons. For example, the non-linearity of the permeability affects in fact on the calculations. In addition, the corners of the core add some errors to the calculation of the mean path length of the core. In the other hand, the cross-sectional area effect of the air gap is larger than the cross-sectional effect of the iron core, which is widely known as fringing effect.

2.2.1 Type of losses in transformer

The losses types of the transformer is mentioning here, even if they are not relative to the core as in coils. The names of them in FEMMT should be also mentioned here.

2.2.1.1 Leakage flux in core

The permeability of the ferromagnetic materials lies between 2000 to 6000, which is opposed to the constant permeability assumption, because there is a small percentage of the flux gets away from the core into the surrounding air which has a low permeability. This external fluxes is the leakage flux. Therefore, since the magnetization behavior of ferromagnetic materials participate in the flux leakage, the core must be handled in the unsaturated region.

2.2.1.2 Hysteresis losses

Due to the normal behavior of alternating current, the flux will increase with the increasing of the current, but it will never reach zero when the current falls again, alternatively a magnetic field is left in the core, which forms a shape of hysteresis loop. A less area of the hysteresis loop can be a way to reduce the hysteresis losses, but it also needs to reduce the applied voltage. This type of losses is related to the iron core.

2.2.1.3 Eddy current losses

The voltage induced in turns of coil, according to Faraday's law, causes some swirls current following in the core; which is widely known as eddy currents. Depending on the larger induced voltage and the low resistivity of the material, the effect and the size of eddy currents will grow proportionally. Therefore, a small strips or laminations and a high resistivity of the material in the design of the core can reduce these current swirls. These losses are also related to the core. A core losses can be uses referring to hysteresis losses and eddy current losses.

2.2.1.4 winding losses

Since not all of the fluxes can pass from the primary winding to the other windings, some winding losses are presented. These losses can be divided into a mutual flux and a small leakage flux as shown in Fig. 2.2. The mutual flux is the remaining flux within the core which links the both windings and the small leakage flux is the passing flux into the air through the primary or the secondary winding. These upcoming equations describe the winding losses. The total flux linkage depends not only on the number of turns of coils, but also on the position of the turns on the coil, which means that the construction of the transformer contribute in reducing the leakage flux.

$$\phi_p = \phi_M + \phi_{LP} \tag{2.9}$$

$$\phi_s = \phi_M + \phi_{LS} \tag{2.10}$$

where ϕ_p and ϕ_s are the total average primary and secondary leakage fluxes, ϕ_{LP} and ϕ_{LS} are the primary and secondary leakage fluxes, and ϕ_M is the mutual flux.

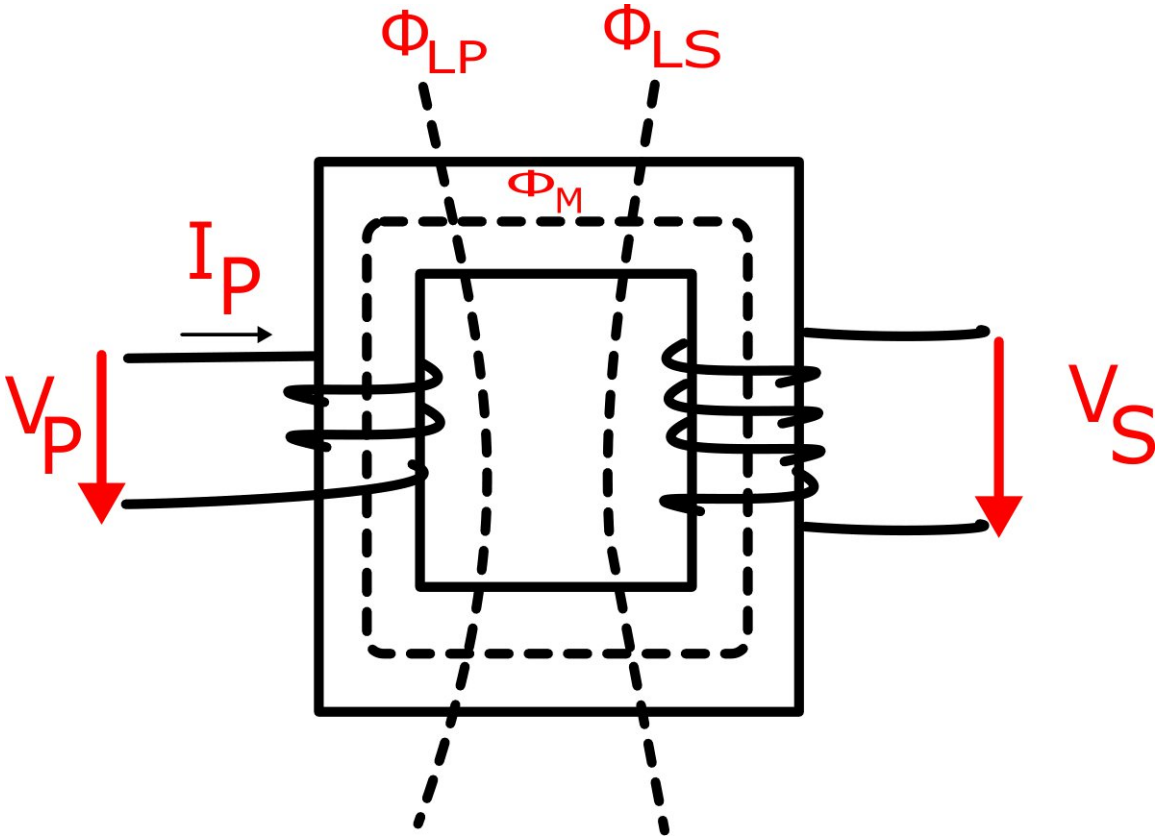

Fig. 2.2: Mutual and leakage fluxes in a transformer core

The voltage ratio across the transformer can be derived from equations 2.9 and 2.10 depending on Faraday's law as

$$\begin{aligned} v_P(t) =& N_P \frac{d\phi_p}{dt} \\ &= N_P[\frac{d\phi_M}{dt} + \frac{d\phi_{L_P}}{dt}] \end{aligned} \tag{2.11}$$

$$\begin{aligned} v_S(t) =& N_S \frac{d\phi_s}{dt} \\ &= N_S[\frac{d\phi_M}{dt} + \frac{d\phi_{L_S}}{dt}] \end{aligned} \tag{2.12}$$

Because the primary voltage equals to the secondary voltage due to the mutual flux, the ratio between two voltages, or transformer ratio, can be approximated if ϕ_M is large enough than ϕ_{LP} and ϕ_{LS}, which is

$$\frac{v_P(t)}{V_S(t)} = \frac{N_P}{N_S} = n \tag{2.13}$$

2.2.2 Equivalent circuit for a real 2-winding transformer

The equivalent circuit of a transformer should contain all of these losses, also the copper losses. From Fig 2.3, the R_C connected to the primary represents the core losses, and the inductance X_M represents the magnetization current, which are proportional to the applied voltage. R_P and R_S represent the resistance of the primary and the secondary winding respectively. X_P and X_S represent the inductance of the primary and the secondary winding respectively. It is very useful to refer the secondary side in the equivalent circuit to the primary side or the primary side to the secondary side, in order to be easily handled [1].

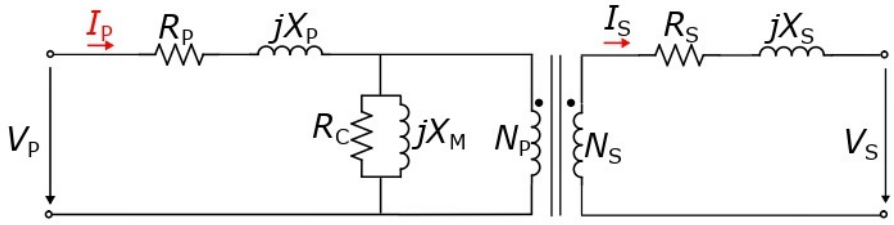

Fig. 2.3: The model of a real 2-winding Transformer

2.2.3 Needing for several winding in a transformer

One of the simplest logical benefits of adding an additional winding to the transformer is to make the transformer multitask, as it can step up and step down time-varying voltages and currents. This can be done by controlling the number of the secondary-turns and the tertiary-turns, which means that one transformer can feed many equipment as needed.

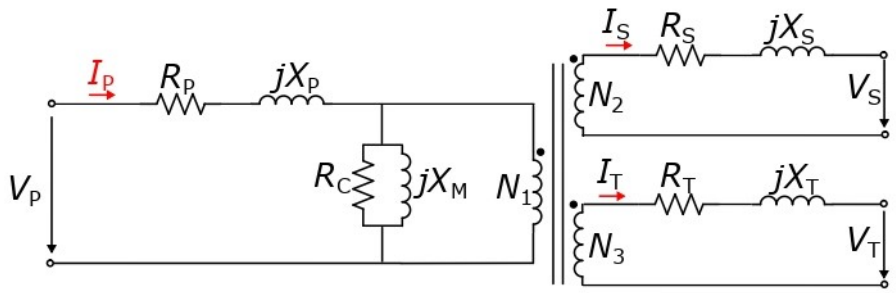

Fig. 2.4: The model of a real 3-winding Transformer

The additional winding can be also used as adjutant winding for supplying lights, fans and so on. It is also very popular to use the third winding in the high voltage measuring test of a transformer. In addition, one of the significant feature of the third winding is to reduce the harmonic caused by the primary and secondary winding if the third winding is connected in delta.

In the field of power electronics, a third winding can be used in conjunction with a resonant circuit to provide high-frequency power conversion. This may be useful in high-efficiency applications. It can also be used as a feedback winding to provide feedback to the controller. In addition, the minimizing of the leakage inductance is required in some applications, such as in resonant converter typologies so adding a third winding can be used to control the leakage inductance of the transformer. It can also be used to provide a DC biasing for some power electronic devices such as transistors or diodes. Fig. 2.4 shows the equivalent circuit of three winding transformer. R_T and X_T represent the resistance and the inductance of the tertiary winding respectively [2].

3 FEM Simulation Framework: FEMMT and FEMM4.2

3.1 Principle of FEMMT

The FEM Magnetic toolbox (FEMMT) is an open source and in-progress project, which can offer the opportunity to work out deal with different points along the process of the designing of magnetic components.FEMMT is advanced in open source with GitHub, allows software teams managing changes to source code over time; which is known scientifically as source control system. FEMMT can do all FEM simulation with "Open Numerical Engineering LABoratory" (ONELAB) which is the connecting of the "Three dimentional finite element mesh generator with built-in pre- and post processing facilities" (GMSH) to and "General Environment for the Treatment of Discrete Problems" (GetDP). The main feature of FEMMt is to attempt an interface to this FEM framework without deviation from python[3].

As several common core types, such as "PQ", "RM", and "EI" core geometries, in single-phase problems can be suitable in a 2D axisymmetric simulation, the fast calculation time in a 2D mesh is the usefulness of this simulation. Consequently, because of this method, many iterations on parameter vectors for optimization problems are granted. 2D axisymmetric view can be understood easily from Fig. 3.1.

Overall, the user can easily control some values passing to the interface without specifying any geometric details. For example, the air gaps values, the winding window height and depth, the insulation distances, the virtual winding windows, the winding types and schemes, and the conductor arrangement would be easily setting.

3.1.1 Structure of FEMMT

The structure of FEMMT is built of python files and non-python text files, where the non-python text files are in reference to the simulation scripts, such as the solvers and some material details. Furthermore, these files are expressed in the finite element solver GetDP; which is an environment to deal with such problems of various dimensions and time states. FEMMT can first do the achievement of the FEM simulation in the ONELAB

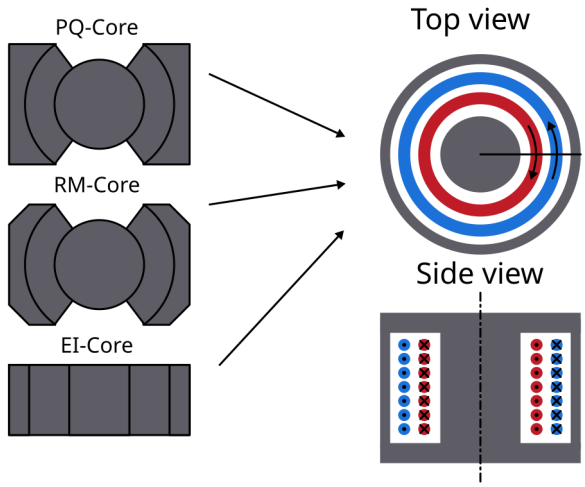

Fig. 3.1: 2D axisymmetric view of different core types

via an Application Programming Interface (API); through which the interaction process takes place with the mesh generator GMSH, knowing that the ONELAB (Python) module connect GMSH and GetDP virtually. Moreover, inside that module the mesh and solver files are sent to a sub client which run the simulation.

As the data files types play a vital role in the FEMMT, they should be mentioned. The solver files are identified with the extension of".Pro" and the mesh files with ".msh". The files, that are responsible about the field results, end with ".pos". The files, that are responsible about the integrated field results, end with ".dat". The field results can be expressed as the distribution of the magnetic flux density and loss density. The integrated field results can be any quantity; that is important to the field results, such as the flux linkage value, different losses values in the core or in the windings, and inductance values.

However, in FEMMT the file "Parameter.Pro" is the communication between the solver scripts and python. It can show the variable parameter values for every simulation run. For example, type of the flag, the frequency, resistances, phases, number of conductors, the voltage across the windings, and the permeability value can be shown[4].

Moreover, the "solver.pro" , "fields.pro", and "values.pro" are included in the major GetDP file is "ind_axi_python_controlled.pro". The "ind_axi_python_controlled.pro" the responsible file for building the description of the problems. Some particular information and data of the problem, such that the geometry, physical numbers, and boundary conditions followed by the definition of the resolution method, some equations, and related objects are also defined in this file. In the file "solver.pro" the equations that have to built are mathematically solved. The simple structure of FEMMT is shown in Fig. 3.2.

However, the entire magnetic toolbox consists of several files; that are related to the materials interpretation, static functions, and pre-simulation scripts[5].

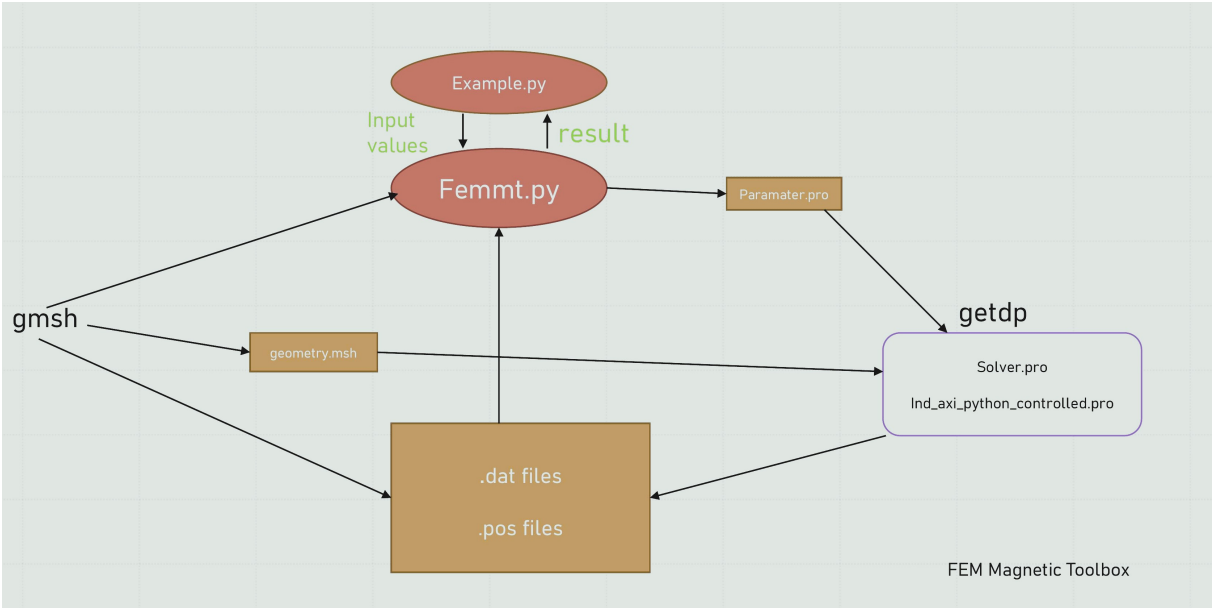

Fig. 3.2: The structure of FEMMT

3.2 Principle of FEMM4.2

FEMM 4.2 is an open source software that can solve electromagnetic problems with low frequency on 2D planer and axisymmetric domains. The name stands for Finite Element Method Magnetics. FEMM 4.2 can deal with linear, or non-linear problems, differently if these problems are magnetostatic, time-harmonic magnetic, electrostatic, or steady-state heat flow. FEMM 4.2 is widely used, freely available for download and use, and its open source nature allows users to modify and extend its capabilities to suite their needs.

3.2.1 Structure of FEMM4.2

FEMM4.2 consists of several components that work together to provide a user interface and powerful simulation. It has a graphical user interface (GUI) to provide an environment for designing and analysing electromagnetic devices. The GUI allows users to create and modify geometries, define material properties, specify boundary conditions, and set up simulation parameters. FEMM 4.2 uses also a solver to solve the electromagnetic field equations. The solver uses the finite element method to discretize the geometry into small elements, and then solve the equations using matrix algebra. The solver can solve a wide range of electromagnetic problems, including static, steady state, and transient problems. FEMM4.2 has a result viewer that allows users to visualize and analyze the results of the simulation [6].

4 Get Inductance In FEMMT

The approximated values of inductances and resistances can be determined in the transformer model through an open and short circuits. An open circuit refers to the situation where the secondary and tertiary windings are open-circuited. Short circuiting a transformer refers in principle to connecting the secondary and the tertiary windings in series. In FEMMT, the function *get_inductance*() is used to find the values of the inductances in a transformer using open circuit tests. The analyzing of three-winding starts from Fig. 4.1. It is already existed only for two windings transformer, so it will be extended for three windings transformer.

4.1 Analysis for Three Windings Transformer

The three windings transformer have the number of turns N_P, N_S, and N_T as well as the self-inductances L_{11}, L_{22}, and L_{33}. the leakage fluxes are designated as ϕ_{L1}, ϕ_{L2}, and ϕ_{L3}. The integral form of Faraday's law of induction is a starting point to find to determine

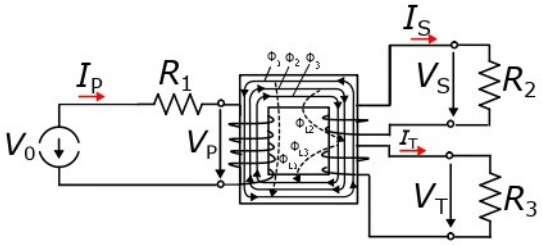

Fig. 4.1: Three winding transformer

the values of inductances, which is

$$IR = \oint_L E \cdot dl = -\frac{d\phi}{dt} \tag{4.1}$$

Applying equation(4.1) for a three windings transformer provides the following relationships

$$-V_0 + R_1 I_P = -\frac{d\phi_1}{dt} = -\frac{d[(N_1\phi_1 - N_1(\phi_2 - \phi_{L1}) - N_1(\phi_3 - \Phi_{L3})]}{dt} \tag{4.2}$$

$$R_2 I_S = -\frac{d\phi_2}{dt} = -\frac{d[N_2(\phi_1 - \phi_{L1}) + N_2\phi_2 - N_2(\phi_3 - \Phi_{L3})]}{dt} \tag{4.3}$$

$$R_3 I_T = -\frac{d\phi_3}{dt} = -\frac{d[N_3(\phi_1 - \phi_{L1}) - N_3(\phi_2 - \phi_{L2}) + N_3\phi_3]}{dt} \tag{4.4}$$

By rearranging these equations, the following expressions are obtained, where the entire flux linked with the respective winding is in the square bracket on the right-hand side of the equation. It is assumed that the flux ϕ_1 passes through all N_1 turns of the primary winding, the flux Φ_2 passes through all N_2 turns of the secondary winding, and the flux ϕ_3 passes through all N_3 turns of the tertiary winding

$$\begin{aligned} V_0 &= R_1 I_P + \frac{d(N_1\phi_1)}{dt} - \frac{d(N_1(\phi_2 - \phi_{L2}))}{dt} - \frac{d(N_1(\phi_3 - \phi_{L3}))}{dt} \\ &= R_1 I_P + \frac{d\phi_{11}}{dt} - \frac{d\phi_{12}}{dt} - \frac{d\phi_{13}}{dt} \end{aligned} \tag{4.5}$$

$$\begin{aligned} 0 &= R_2 I_S - \frac{d(N_2(\phi_1 - \phi_{L1}))}{dt} + \frac{d(N_2\phi_2)}{dt} - \frac{d(N_2(\phi_3 - \phi_{L3}))}{dt} \\ &= R_2 I_S - \frac{d\phi_{21}}{dt} + \frac{d\phi_{22}}{dt} - \frac{d\phi_{23}}{dt} \end{aligned} \tag{4.6}$$

$$\begin{aligned} 0 &= R_3 I_T - \frac{d(N_3(\phi_1 - \phi_{L1}))}{dt} - \frac{d(N_3(\phi_2 - \phi_{L3}))}{dt} + \frac{d(N_3\phi_3)}{dt} \\ &= R_3 I_T - \frac{d\phi_{31}}{dt} - \frac{d\phi_{32}}{dt} + \frac{d\phi_{33}}{dt} \end{aligned} \tag{4.7}$$

These equations describe the relationship between the voltages induced in the primary, secondary, and tertiary windings of a transformer and the time derivative of the fluxes linking these windings. Because $L = \frac{\phi}{I}$, the previous equations can be written as

$$V_0 = R_1 I_P + L_{11}\frac{dI_P}{dt} - L_{12}\frac{dI_S}{dt} - L_{13}\frac{dI_T}{dt} \tag{4.8}$$

$$0 = R_2 I_S - L_{21}\frac{dI_P}{dt} + L_{22}\frac{dI_S}{dt} - L_{23}\frac{dI_T}{dt} \tag{4.9}$$

$$0 = R_3 I_3 - L_{31}\frac{dI_P}{dt} - L_{32}\frac{dI_S}{dt} + L_{33}\frac{dI_T}{dt} \tag{4.10}$$

Therefore, there will be sex independent values to determine the parameters in the equivalent circuit diagram of three-winding transformer as shown in Fig. 4.2. The previous equations can be described by the following matrix; which represents a system of these equations

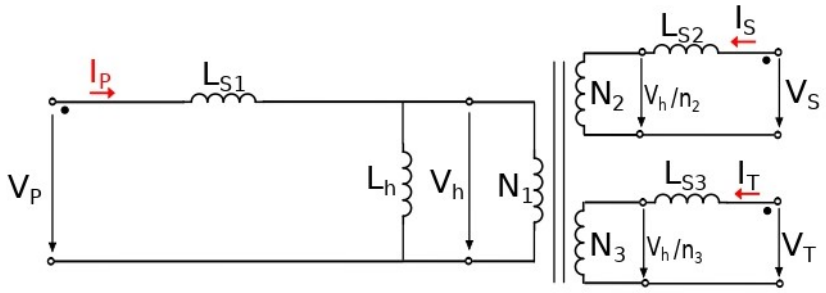

Fig. 4.2: Equivalent circuit diagram

$$\begin{bmatrix} \mathrm{V}_P \\ V_S \\ V_T \end{bmatrix} = \begin{bmatrix} L_{11} & M_{12} & M_{13} \\ M_{12} & L_{22} & M_{23} \\ M_{13} & M_{23} & L_{33} \end{bmatrix} * \frac{d}{dt} \begin{bmatrix} I_P \\ I_S \\ I_T \end{bmatrix} \tag{4.11}$$

The inductance matrix is a square matrix that describes the inductances and mutual inductances between the windings of a three-winding transformer. The diagonal terms of the matrix represent the self-inductances of each winding (i.e., the inductance of each winding with respect to itself), while the off-diagonal terms represent the mutual inductances between the windings (i.e., the inductance between two windings induced by the magnetic field produced by the current in the other winding). In the equation, L_{11}, L_{22}, and L_{33} correspond to the self-inductances of the primary, secondary, and tertiary windings, respectively. The off-diagonal terms, M_{12}, M_{13}, and M_{23}, represent the mutual inductances between the primary and secondary, primary and tertiary, and secondary and tertiary windings, respectively. The mutual inductances are typically denoted with the letter M, while the self-inductances are typically denoted with the letter L. In a three-winding transformer, the mutual inductances M_{12} and M_{21}, M_{13} and M_{31}, M_{23} and M_{32} are equal due to the symmetry of the transformer The derivative on the right-hand side of the equation represents the rate of change of the currents in the three windings. This derivative is calculated with respect to time, and it describes how the currents in each winding change over time. The equations derived from the Kirchhoff's laws and the inductance matrix can be used to find the values of Lh, $Ls1$, $Ls2$, and $Ls3$ for a three-winding transformer. Lh represents the main inductance on the primary side, and it can be found using this equation

$$L_h = \frac{M_{12}M_{13}}{M_{23}} \tag{4.12}$$

$Ls1$, $Ls2$, and $Ls3$ represent the self-inductances on the other side, and they can be found using the equations

$$L_{s1} = L_{11} - \frac{M_{12}M_{13}}{M_{23}} \tag{4.13}$$

$$L_{s2} = L_{22} - \frac{M_{12}M_{23}}{M_{13}} \tag{4.14}$$

$$L_{s3} = L_{33} - \frac{M_{13}M_{23}}{M_{12}} \tag{4.15}$$

4.1.1 Couple factors

The coupling factors are the ratios between the mutual inductances and the self-inductances of the windings in a transformer. They are also referred to as coupling inductances. In a transformer with multiple windings, the coupling factors determine the behavior and transmission characteristics of the transformer. They also affect the performance and efficiency of the transformer.

The coupling factors can be calculated by dividing the mutual inductances by the square roots of the corresponding self-inductances, or can be calculated by dividing the mutual fluxes by the leakage fluxes. The coupling factors between the primary and secondary winding, primary and tertiary winding, and secondary and tertiary winding are denoted as k_{12}, k_{13}, and k_{23}, respectively. These equations are

$$K_{12} = K_{21} = \sqrt{K_{12}K_{21}} = \frac{\phi_{12}}{\phi_{11}} = \frac{M_{12}}{\sqrt{L_{11}L_{22}}} \tag{4.16}$$

$$K_{13} = K_{31} = \sqrt{K_{13}K_{31}} = \frac{\phi_{13}}{\phi_{33}} = \frac{M_{13}}{\sqrt{L_{11}L_{33}}} \tag{4.17}$$

$$K_{23} = K_{32} = \sqrt{K_{23}K_{32}} = \frac{\phi_{23}}{\phi_{33}} = \frac{M_{23}}{\sqrt{L_{22}L_{33}}} \tag{4.18}$$

The coupling factors play an important role in the design of transformer windings and circuits. When selecting transformer windings, the ratio of coupling factors must be chosen to correspond to the desired transmission characteristics and operating conditions.

4.1.2 Applying these Equations in FEMMT

To prepare for the inductance calculation of a three-winding transformer, the frequencies, currents, and phases need to be defined. Additionally, the previous inductance logs are removed to ensure that new inductance values are generated. Finally, the fluxes induced in each winding are extracted from the corresponding *flux_linkage.dat* files. listing 4.1 shows that the "frequencies" list contains a single value, "*op_frequency*" which is presumably the operating frequency of the transformer. In addition, The "currents" list contains three sub-lists, each representing the current in one of the windings. Each sub-list contains three values, with only one of them being non-zero. This indicates that the transformer is being tested with single-phase current, where only one of the windings is energized at a time. The "phases" list contains three sub-lists, each representing the phase angle between the current and voltage in one of the windings. In this case, all of the sub-lists have the same values, indicating that the current and voltage are in phase for all of the windings.

```
if len(self.windings) == 3:
    .
    .
    .
    if self.valid:
```

```
        frequencies = [op_frequency] * 3
        currents = [[I0, 0, 0], [0, I0, 0], [0, 0, I0]]
        phases = [[0, 180, 180], [0, 180, 180], [0, 180, 180]]
```

Listing 4.1: Main Parameters

Listing 4.2 shows that each of the "*Flux_Linkage.dat*" contain data about the fluxes induced in each winding by the current flowing through the other windings. The values of these fluxes are assigned to variables Phi_{ij}, where i denotes the winding in which the flux is induced and j denotes the winding through which the current is flowing. Finally, the values of these fluxes are printed to the console using formatted strings

```
        .
        .
        # with opening ("Flux_Linkage_i.dat")files:
        # Fluxes induced in Winding 1
            Phi_11 = float(line[0].split(sep=' ')[2])
            Phi_12 = float(line[1].split(sep=' ')[2])
            Phi_13 = float(line[2].split(sep=' ')[2])

        # Fluxes induced in Winding 2
            Phi_21 = float(line[0].split(sep=' ')[2])
            Phi_22 = float(line[1].split(sep=' ')[2])
            Phi_23 = float(line[2].split(sep=' ')[2])

        # Fluxes induced in Winding 3
            Phi_31 = float(line[0].split(sep=' ')[2])
            Phi_32 = float(line[1].split(sep=' ')[2])
            Phi_33 = float(line[2].split(sep=' ')[2])
```

Listing 4.2: Induced Fluxes

Listing 4.3 shows the calculations of the coupling factors for a three-winding transformer using the fluxes induced in each winding. The factors between winding 1 and 2, 1 and 3, and 2 and 3 are denoted by $K21$, $K31$, $K12$, $K32$, $K13$, and $K23$, respectively. Then, it simplifies the coupling factors between each pair of windings, denoted by k_1, k_2, and k_3 as $K_{ij} = K_{ji}$. The coupling factors are calculated as the square root of the product of the corresponding $K_{ij}K_{ji}$.

```
        .
        .
        # Coupling Factors
        K_21 = Phi_21 / Phi_11
        K_31 = Phi_31 / Phi_11
        K_12 = Phi_12 / Phi_22
        K_32 = Phi_32 / Phi_22
        K_13 = Phi_13 / Phi_33
        K_23 = Phi_23 / Phi_33

        k_1 = (K_21 * K_12) ** 0.5
        k_2 =  (K_31 * K_13) ** 0.5
```

```
13          k_3 =  (K_23 * K_32) ** 0.5
```

Listing 4.3: Couple factors

Listing 4.4 shows the calculation of the mutual inductances between the windings of a three-winding transformer based on the previously calculated coupling factors. The main mutual inductances are calculated using the coupling factors and the self-inductances of the corresponding windings. Additionally, the code also calculates the mutual inductances using the ideal values ($M_{12} = M_{21}, M_{13} = M_{31}, M_{23} = M32$) to verify the correctness of the calculated values.

```
1           .
2           .
3           # Main/Counter Inductance
4           # With opening (L_i_j.dat) files:
5           self.M_12 = k_1 * (self.L_1_1 * self.L_2_2) ** 0.5
6           self.M_13 = k_2 * (self.L_1_1 * self.L_3_3) ** 0.5
7           self.M_23 = k_3 * (self.L_2_2 * self.L_3_3) ** 0.5
8           M_12 = self.L_1_1 * K_21
9           M_21 = self.L_2_2 * K_12
10          M_13 = self.L_1_1 * K_31
11          M_31 = self.L_3_3 * K_13
12          M_23 = self.L_2_2 * K_32
13          M_32 = self.L_3_3 * K_23
```

Listing 4.4: Mutual inductances

Listing 4.5 shows the used equations to calculate the inductances for a three-winding transformer in the equavilant circuit in Fig. 4.2. The leakage inductance values are determined using the mutual inductances ($M_{12}, M_{13}, and M_{23}$) that were calculated in the previous step. In particular, the equations show that the self-inductances of each secondary winding are reduced by the influence of the other two windings. The main or high-voltage primary inductance is calculated using the mutual inductances between the primary and the other two windings.

```
1           .
2           .
3           L_s1 = self.L_1_1 - (self.M_12 * self.M_13) / self.M_23
4           L_s2 = self.L_2_2 - (self.M_12 * self.M_23) / self.M_13
5           L_s3 = self.L_3_3 - (self.M_13 * self.M_23) / self.M_12
6           L_h = (self.M_12 * self.M_13) / self.M_23
```

Listing 4.5: leakage inductances in the equivalent circuit 4.2

Table 4.1 compares the results of using FEMMT and FEMM4.2 for finding the inductance of a three-winding transformer for different types of conductors. The table provides information on the number of turns, current, and frequency used in the simulation. It also shows the results of the simulation for the primary, secondary, and tertiary inductance using both FEMMT and FEMM4.2 models.The table shows that the inductance values obtained for each type of conductor also differ between FEMMT and FEMM4.2

models. Based on the results, it can be seen that the inductance values obtained using FEMMT and FEMM4.2 models differ for all three types of conductors. The differences in the inductance values obtained using the two models could be due to the differences in the underlying algorithms, numerical methods, and mesh used by the models for solving electromagnetic problems [7].

Tab. 4.1: Comparison of *get_inductance* results for three-winding transformer using FEMMT and FEMMT4.2

Characteristic of the transformer		
No. of turns	8, 6, 12	
current (I0)	8 Amp	
frequency	250000 Hz	
type of model	FEMMT	FEMM4.2
type of conductor	Round-Solid	
primary Inductance	$3.373 * 10^{-}5$ H	$3.464.10^{-}5$ H
secondary Inductance	$1.932 * 10^{-}5$ H	$1.984 * 10^{-}6$ H
tertiary Inductance	$7.523 * 10^{-}5$ H	$1.724 * 10^{-}5$ H
type of conductor	Round-Litz	
primary Inductance	$3.464 * 10^{-}5$ H	$3.565.10^{-}5$ H
secondary Inductance	$1.953 * 10^{-}5$ H	$2.009 * 10^{-}5$ H
tertiary Inductance	$7.623 * 10^{-}5$ H	$7.862 * 10^{-}6$ H
type of conductor	Solid-Litz	
primary Inductance	$3.403 * 10^{-}5$ H	$3.486.10^{-}5$ H
secondary Inductance	$1.950 * 10^{-}5$ H	$2.009 * 10^{-}5$ H
tertiary Inductance	$7.562 * 10^{-}5$ H	$7.802 * 10^{-}6$ H

5 Multiple Winding Transformer (n-transformer)

The current code in FEMMT was designed to handle inductors and two-winding transformers using the *Flag_Inductor* and *Flag_Transformer* flags. The *Flag_Three_Transformer* was added to handle a three-winding transformer, but it was primarily for testing and understanding the project as shown in listing 5.1. These flags are used to determine how to perform certain calculations or simulations based on the type of component being simulated and the number of winding that it has. They are used in "Parameter.Pro" file; which is the connection between solver scripts and python

To develop a new code that can handle transformers with any number of windings, it is necessary to replace the existing if statements in the code with a dynamic "for-loop". The use of a "for-loop" will allow the program to iterate through each winding in the transformer and define the necessary variables, making the code more adaptable and versatile. The flags will not be needed in future to determine the type of the component as shown in listing 5.2.

Since the number of windings can vary for different transformers, it will be necessary to modify the lists in the code to be dynamic. This will allow the code to handle a variable number of windings, with the number of iterations for the for-loop being determined by the length of the winding list. These changes were also made to the "solver.pro" "*ind_axi_python_controlled.pro*", "fields.pro", and "values.pro" files.

```
        .
        # Magnetic Component Type
        if self.component_type == ComponentType.Inductor:
            text_file.write(f"Flag_Transformer = 0;\n")
            text_file.write(f"Flag_Three_Transformer = 0;\n")
        if self.component_type == ComponentType.Transformer:
            text_file.write(f"Flag_Transformer = 1;\n")
            if len(self.windings) == 3:
                text_file.write(f"Flag_Three_Transformer = 1;\n")
            else:
                text_file.write(f"Flag_Three_Transformer = 0;\n")
        if self.component_type == ComponentType.IntegratedTransformer:
```

```
            text_file.write(f"Flag_Transformer = 1;\n")
            if len(self.windings) == 3:
               text_file.write(f"Flag_Three_Transformer = 1;\n")
            else:
               text_file.write(f"Flag_Three_Transformer = 0;\n")
```

Listing 5.1: old code using flags

By using dynamic lists, the code can be modified to handle transformers with any number of windings, and the for-loop can be used to define the necessary variables for each winding. This will make the code more flexible and adaptable to different types of transformers, which can be useful in a variety of applications.

Overall, by replacing "if statements" with a dynamic "for-loop" and modifying lists to be dynamic, the code can be modified to handle transformers with any number of windings, making it more versatile and adaptable to different applications.

```
        .
        .
        # Magnetic Component Type
        if self.component_type == ComponentType.Inductor:
           text_file.write(f"Number_of_Windings = {len(self.windings)};\
    n")
        if self.component_type == ComponentType.Transformer:
           text_file.write(f"Number_of_Windings = {len(self.windings)};\
    n")
        if self.component_type == ComponentType.IntegratedTransformer:
           text_file.write(f"Number_of_Windings = {len(self.windings)};\
    n")
```

Listing 5.2: new code using number of windings

An illustrative example is shown in listing 5.3 and listing 5.4. The listing 5.3 shows the old code of three-winding transformer to define physical numbers of conductors iCOND1, iCOND2, and iCOND3, which represent different regions in the simulation. These numbers are used to identify and manipulate different parts of the simulation in the "*ind_axi_python_controlled.pro*". The listing 5.4 shows the developed code for n-winding transformer, that can define physical numbers iCOND1, iCOND2, iCOND3, etc.

```
        .
        .
        iCOND1              = 130000;
        istrandedCOND1      = 140000;

        If(Flag_Transformer)
            iCOND2            = 131000;
            istrandedCOND2    = 141000;
        EndIf

        If(Flag_Three_Transformer)
            iCOND3            = 132000;
            istrandedCOND3    = 142000;
```

```
14          EndIf
```

Listing 5.3: example using if-statement

```
1           .
2           .
3           For n In {1:n_windings}
4           iCOND~{n} = 130000 + 1000*(n-1);
5           istrandedCOND~{n} = 140000 + 1000*(n-1);
6           EndFor
```

Listing 5.4: example using for-loop

5.1 Virual Winding Window

The winding window is a 2D representation of the 3D rotated winding window that exists for every FEMMT model. The method provided in FEMMT allows the user to split a virtual winding window into smaller windows based on a split type and split factors. The split type determines the arrangement of the virtual winding windows after splitting, and the split factors are values between 0 and 1 that determine a horizontal and vertical line at which the window is split. This method can create up to four virtual winding windows based on the split type and split factors. The four virtual winding windows are[8]:

1. NoSplit: Returns a single virtual winding window with the same size as the real winding window.
2. HorizontalSplit: Returns two virtual winding windows, one for the top and one for the bottom part. The height of the splitting line can be adjusted using a "$horizontal\_split\_factor$" (a value between 0 and 1).
3. VerticalSplit: Returns two virtual winding windows, one for the left and one for the right part. The position of the splitting line can be adjusted using a "$vertical\_split\_factor$" (a value between 0 and 1).
4. HorizontalAndVerticalSplit: Returns four virtual winding windows, one for each corner in the order of "top_left", "top_right", "$bottom\_left$", and "$bottom\_right$". The horizontal and vertical split factors can be used to adjust the sizes of each grid cell.

The "$TenCells\_Split$" method is a new method and a specific implementation of n-winding transformer that can divide the winding window into 10 virtual winding windows. These virtual winding windows are used to draw the conductors for 10 windings, as the program appears to only set a single conductor to the current virtual winding window. The splitting points are determined based on the values of "$horizontal\_split\_factor$", "$horizontal\_split\_factor_2$", "$horizontal\_split\_factor\_3$, "$horizontal\_split\_factor\_4$", and "$vertical\_split\_factor$".

5.2 Simulation of ten-winding Transformer

The results of a single frequency sweep simulation appears in Fig. 5.1 . The simulation was done at a frequency of 250 kHz. There are ten windings in the simulation, with different numbers of turns. For each winding, the object contains several parameters, that can be shown in ".json" file:

1. Turn losses
2. Winding losses
3. Flux linkage
4. Flux over current
5. Voltage
6. Current
7. Real power
8. Reactive power
9. Apparent power

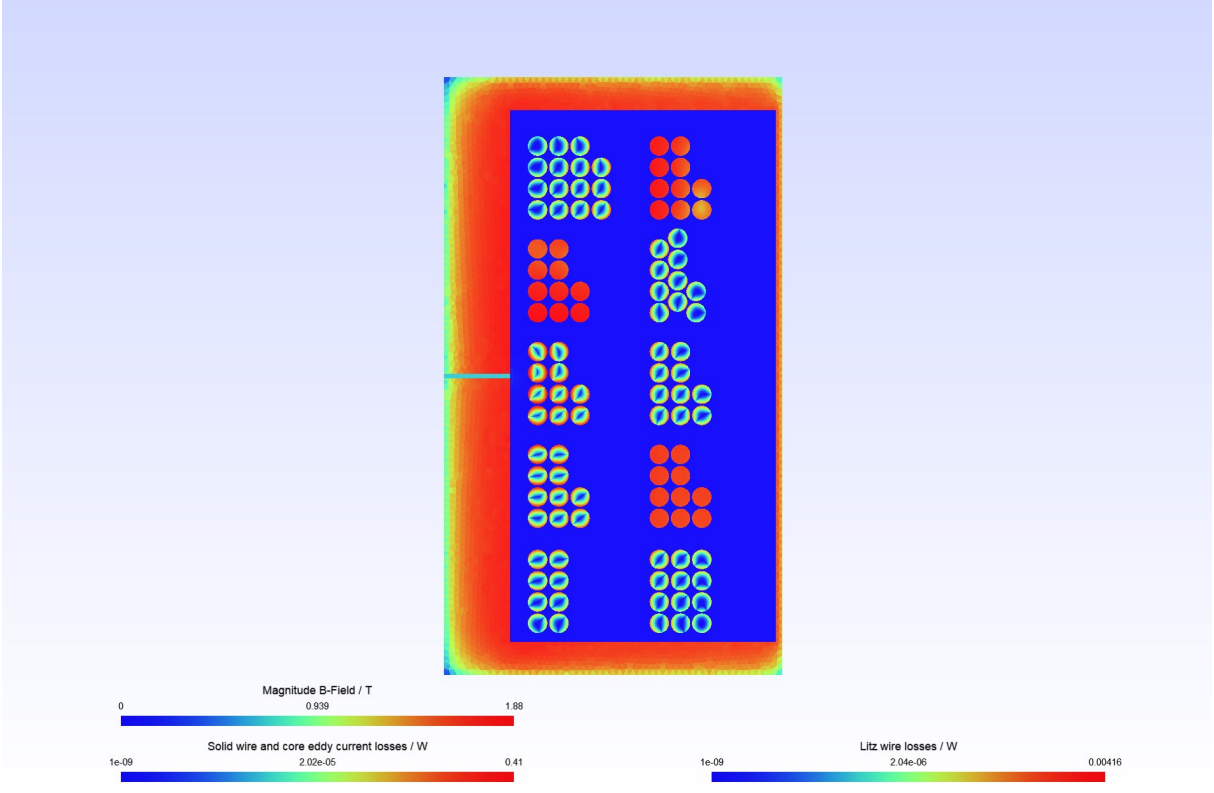

Fig. 5.1: Simulation of ten-winding transformer

6 Conclusion

The simulation results obtained from FEMMT for the three-winding transformer were found to be in good agreement with those obtained from FEMM 4.2. The code was then provided to be valid for n-windings. This indicates that the developed code for n-winding transformer in FEMMT is valid and can simulate transformers with any number of windings. The use of dynamic lists and a for-loop in the code has made it more versatile and adaptable to different types of transformers, which can be useful in a variety of applications. The development of the "$TenCells\_Split$" method also adds more flexibility to the program, allowing the user to divide the winding window into 10 virtual winding windows. Overall, the extension of FEMMT project with the capability of building n-windings and validating the simulation of n-winding transformer has proven to be a success.

Bibliography

[1] S. J Chapman, *Electric machinery fundamentals*. McGraw-hill, 2004.

[2] D. W. Hart and D. W. Hart, *Power electronics*. McGraw-Hill New York, 2011, vol. 166.

[3] *ONELAB*, https://www.onelab.info/, Visited: 2022-10-25.

[4] T. Piepenbrock, "Automated fem transformer design for a dual active bridge," *Masterarbeit. Paderborn: Universität Paderborn*, 2021.

[5] N. Förster, T. Piepenbrock, P. Rehlaender, O. Wallscheid, F. Schafmeister, and J. Boecker, "An open-source fem magnetics toolbox for power electronic magnetic components," in *PCIM Europe 2022; International Exhibition and Conference for Power Electronics, Intelligent Motion, Renewable Energy and Energy Management*, VDE, 2022, pp. 1–10.

[6] *FEMM*, https://www.femm.info/wiki/Download, Visited: 2022-01-20.

[7] M. Albach, *Induktivitäten in der Leistungselektronik*. Springer, 2017.

[8] *LEA Python Toolbox*, https://github.com/upb-lea/FEM_Magnetics_Toolbox, Visited: 2022-10-25.