Electromagnetic Design of Induction Motor Supporting Arbitrary Poles, Phases, and Slot Geometries

Rajagopal.B
3rd Year B.Tech. – EEE
Vellore Institute of Technology – Vellore

SUMMER INTERNSHIP REPORT

Submitted to:

Dr. Jose Titus
Assistant Professor
EE Department
Indian Institute of Technology – Hyderabad

Academic Year: 2024 – 2025

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY
HYDERABAD – 502284

June 2025

Acknowledgement

I would like to express my sincere gratitude to **Dr. Jose Titus**, Assistant Professor, Department of Electrical Engineering, Indian Institute of Technology, Hyderabad, for his invaluable mentorship and guidance throughout the course of this project. His technical insights, continuous support, and encouragement were instrumental in shaping the direction of this work and enhancing my understanding of electromagnetic machine design.

I am also thankful to the faculty,staff and Lab members of the Department of Electrical Engineering at IIT Hyderabad for providing the necessary facilities and fostering a supportive academic environment. Their cooperation greatly contributed to the successful execution of this project.

My heartfelt thanks extend to my friends and peers for their constructive feedback, motivation, and camaraderie, which made this learning journey both intellectually enriching and personally fulfilling.

Sincerely,
Rajagopal.B

BONAFIDE CERTIFICATE

This is to certify that Mr. Rajagopal.B, Roll Number 22BEE0063, a student of the Department of Electrical and Electronics Engineering, Vellore Institute of Technology, Vellore, has successfully completed the summer internship on the topic **"Electromagnetic Design of Induction Motors Supporting Arbitrary Poles, Phases, and Slot Geometries"** under the mentorship of Dr. Jose Titus, Assistant Professor, Indian Institute of Technology, Hyderabad.

This work is an authentic record of the research and design activities carried out by him during the period from 15.05.2025 to 30.06.2025 in partial fulfillment of the requirements for the summer internship program.

Mr. Rajagopal.B has demonstrated consistent effort, technical curiosity, and a high level of commitment throughout the project. The methodology and outcomes presented in this work contribute meaningfully to the field of computational electrical machine design.

I wish him continued success in all his future academic and professional pursuits.

Dr. Jose Titus
Assistant Professor
Department of Electrical Engineering
Indian Institute of Technology, Hyderabad
Date: 30/05/2025

Contents

1 Introduction

1.1 Motivation and Scope

Induction motors are the backbone of modern industry due to their simplicity, robustness, and reliability. Designing an efficient and cost-effective motor typically involves iterative calculations, geometric planning, and simulation—all of which are time-consuming when done manually. Most existing design tools are limited to fixed pole-slot combinations, predefined slot shapes, or lack user control over materials and geometrical flexibility.

This project aims to overcome those limitations by developing a generalized, automated design model for induction motors. The model can generate accurate electromagnetic designs for arbitrary numbers of poles, phases, and slot geometries while allowing the user to input real-world constraints like nameplate data and material choices.

1.2 Problem Statement

Traditional motor design approaches often assume standard configurations and require manual recalculations when any specification changes. This becomes inefficient when dealing with custom applications, educational case studies, or research-oriented prototypes. Furthermore, many tools lack support for non-standard slot shapes or custom material usage, making them unsuitable for flexible design exploration.

There is a need for a tool that not only automates the fundamental motor design calculations but also supports customization at the input level and adapts automatically to any feasible pole, phase, and slot configuration.

1.3 Objectives of the Project

The main objective of this project is to create a generalized Lua-based script for the automated electromagnetic design of induction motors. Key goals include:

- Supporting user-defined input parameters such as power, voltage, frequency, pole count, and materials.
- Enabling slot geometry selection (e.g., trapezoidal, rectangular) for both stator and rotor.
- Automating the computation of electromagnetic parameters including airgap flux, number of turns, slot dimensions, and tooth widths.
- Ensuring compatibility with arbitrary combinations of pole numbers, slot counts, and phase configurations.
- Generating FEMM-ready geometry with accurate material and coil assignments for simulation.

This tool serves as a modular foundation for engineers, students, and researchers to prototype and evaluate motor designs quickly and accurately.

2 Design Strategy Overview

2.1 User Inputs and Design Constraints

The automated design tool is driven by a flexible set of user inputs. These inputs are grouped into electrical specifications, material selections, geometric settings, and slot configuration options. The model is built to adapt its computations and drawing routines based on these values, enabling support for a wide variety of motor designs.

2.1.1 Electrical Specifications

The user defines basic operating parameters such as:

- Output power
- Supply line voltage
- Frequency
- Number of poles
- Rated Power factor
- Rated Slip
- Efficiency
- Number of phases

These values govern the mechanical speed, airgap power, and input current required, forming the foundation for all subsequent calculations.

2.1.2 Material Selections

Materials can be independently selected for:

- Stator core
- Rotor core
- Stator winding
- Rotor conductor (e.g., squirrel cage)
- Shaft (typically modeled as air for magnetic purposes)

properties directly affect magnetic performance, losses, and slot fill behavior. Users can easily switch between standard grades like M-19 steel, copper, and aluminum.

2.1.3 Winding and Current Density Parameters

The user provides values such as:

- Stator current density
- Surface current density
- Peak Stator Tooth Flux Density
- Winding factor
- Copper fill factor

These determine the number of conductors, the area of each slot, and the magnetic loading. The tool assumes single-layer full-pitch winding, and current per conductor is computed automatically.

2.1.4 Geometry and Slot Configuration

Key geometric inputs include:

- Outer diameter of the stator
- Number of stator and rotor slots
- Desired slot shapes and types
- Tooth top height (bridge height)
- Rotor type (Cage type or Wounded type)

Slot geometry can be selected from pre-defined shapes like rectangular, trapezoidal, or curved-corner variants for both stator and rotor. Slot types (open, semi-closed, closed) can also be chosen based on the design requirement.

2.1.5 Slot Design Ratios

To generalize the tool across machines of different sizes, the following empirical ratios are user-defined:

- Slot height-to-width ratio (dim_factor): Controls the slot's tall or flatness.[3]
- Slot opening-to-base width ratio (dim_factor1): Affects the taper in trapezoidal slots.[4]

These ratios are taken from design recommendations in machine design literature, ensuring manufacturability and thermal viability.The details about this factor is mention as comment in code.

2.1.6 Design Constraints

- Rotor slots are automatically adjusted to avoid cogging and torque ripple issues.
- Slot fill is limited using realistic fill factor and current density constraints.
- Only valid combinations of slots, poles, and phases are allowed.
- Winding logic assumes single-layer, full-pitch coils for simplicity and clarity.

This flexible input structure forms the foundation of a robust, adaptive, and reusable induction motor design tool.

2.2 General Workflow of the Tool

This design tool is implemented using the Lua scripting language, which is directly supported by FEMM (Finite Element Method Magnetics) — a widely used software for electromagnetic analysis of 2D models like motors and transformers.

2.2.1 Why Lua and FEMM?

FEMM allows precise simulation of magnetic fields, and Lua provides a way to automate the entire modeling and analysis process. By writing a Lua script, the motor's geometry, materials, winding layout, and boundary conditions can be generated and solved entirely by code—without using the graphical interface manually.

Lua functions such as `mi_addnode()`, `mi_addmaterial()`, and `mi_analyze()` allow the program to build and simulate motor models based on user-defined inputs.

2.2.2 General Design Flow

The tool follows a clear structure:

- **Inputs:** User provides electrical specs, materials, and slot preferences.
- **Calculations:** The script computes flux density, turns per phase, slot dimensions, and winding layout.
- **Design:** Geometry is automatically drawn in FEMM using Lua commands.
- **Analysis:** FEMM runs the simulation and displays the flux pattern and magnetic performance.

This workflow enables rapid prototyping of motors with different slot types, pole numbers, and materials—making the tool both flexible and practical.

2.3 Design Assumptions and Limitations

2.3.1 Assumptions

- Single-layer, full-pitch winding is used for all designs. Fractional pitch or double-layer configurations are not included to maintain straightforward phase assignment.

- 2D planar geometry is assumed (FEMM limitation), which ignores end effects and assumes uniformity along the axial length.
- Linear magnetic materials are used from FEMM's built-in library. Saturation is considered only through flux density limits, not non-linear B-H curves.
- No skewing of slots or rotor bars is included, though this can affect torque ripple and harmonic content.
- Rotor slot count is auto-chosen to avoid cogging and crawling by checking:

$$S_s - S_r \notin \{\pm P, \pm 2P, \pm 3P, \pm 5P\}[1] \tag{1}$$

- Slot fill factor and current density are used to prevent overdesign, but thermal modeling is not included.

2.3.2 Limitations

- No performance curves (torque-speed, efficiency maps) are generated — only flux plots and field visualization are supported in this version.
- Only predefined slot shapes (rectangular, trapezoidal, curved corners, etc.) are supported. Custom shapes must be manually defined in a separate section.
- No end-winding modeling or loss calculations are performed.
- Mechanical aspects like bearings, frame, or cooling systems are not part of the model.

Despite these limitations, the tool provides accurate electromagnetic design suitable for educational, prototyping, and early-stage industrial applications.

3 Technical Calculations

3.1 Electromagnetic and Geometrical Parameter Calculations

The core of this automation lies in converting electrical nameplate data into precise geometrical dimensions. The script starts by reading key inputs such as output power, voltage, frequency, number of poles, and outer diameter. Using equations from T.A. Lipo's book *"Introduction to AC Machine Design"*, values like stator inner diameter, airgap length, and effective axial length are calculated.

The design process follows these key computational steps with complete references:

1. **Synchronous and Mechanical Speed**

$$N_s = \frac{120 \times f}{P} \quad \omega_m = \frac{2\pi f}{(P/2)} \tag{2}$$

   Source: Basic motor speed equation (Chapter 1, Eq. 1.6, Page 15)[1].
   Determines the fundamental speed characteristics of the motor.

2. **Electromagnetic Torque Calculation**

$$T_e = \frac{P_{out}}{\omega_m} \tag{3}$$

Reference: Eq. 1.6 (Page 20).
Calculates the torque produced from output power and mechanical speed.[1]

3. **Air Gap Flux Density Estimation**

$$B_{g1} = 0.5 \times B_t \tag{4}$$

Reference: Page 271, Eq. (6.42).
Assumes peak airgap flux density is half the tooth saturation density (typically 1.6-2.0T).[1]

4. **Linear Current Density Specification**

$$K_s = K_{s1} \text{ A/m} \tag{5}$$

Reference: Table 6.6 (Page 259) for medium-sized motors.
Typical value for 30kW class motors, affects torque density(Given as input).[1]

5. **Stator Inner Diameter Calculation**

$$\frac{D_{is}}{D_{os}} = \left(\frac{b}{a} + \frac{2K_s}{ak_{cu}J_sD_{os}}\right) - \sqrt{\left(\frac{b}{a} + \frac{2K_s}{ak_{cu}J_sD_{os}}\right)^2 - \frac{1}{a}} \tag{6}$$

Reference: Eq. 6.77 (Page 276)[1].
Where:

- $a = 0.4\text{-}0.6$ (geometric constant, Page 275)[1]
- $b = 0.2\text{-}0.9$ (yoke thickness factor, Page 276)[1]
- $k_{cu} = 0.4\text{-}0.6$ (copper fill factor)
- $J_s = 3\text{-}6 \text{ A/mm}^2$ (current density)

6. **Feasibility Check of Diameter Ratio**

$$\epsilon_{pci} = \frac{3a\left(\frac{D_{is}}{D_{os}}\right)^2 - 4b\left(\frac{D_{is}}{D_{os}}\right) + 1}{\left(\frac{k_{cu}J_s}{4}\right) \cdot \left[\left(\frac{D_{is}}{D_{os}}\right)^2 - a\right]} \tag{7}$$

Reference: Page 277, Section 6.7.2.
Design is feasible if $\epsilon_{pci} \geq 0$ (indicates physically realizable geometry).[1]

7. **Effective Core Length Calculation**

$$l_e = \frac{480\left(\frac{P_{out}}{\omega_{mech}}\right)}{\sqrt{2}\pi^2 k_1 k_{cu} \eta_{gap} \cos\theta_{gap} D_{os}^3 B_{g1} J_s \left[a\left(\frac{D_{is}}{D_{os}}\right)^3 - 2b\left(\frac{D_{is}}{D_{os}}\right)^2 + \left(\frac{D_{is}}{D_{os}}\right)\right]} \tag{8}$$

Reference: Eq. 6.62 (Page 273).
Includes correction factors for winding distribution (k_1) and gap efficiency (η_{gap}).[1]

8. **Airgap and MMF Calculations**

$$g = 0.003 \times \sqrt{P/2} \times \tau_p \quad \text{(Eq. 6.111, Page 293)} \tag{9}$$

$$g_e = K_c \times g \quad (K_c = 2.5 \text{ typical (carter's factor)}) \tag{10}$$

$$F_{p1} = \frac{B_{g1} g_e}{\mu_0} \quad \text{(Eq. 3.77, Page 119)} \tag{11}$$

These determine the magnetic circuit requirements.[1]

9. **Current Components**

$$K_{sm} = \frac{P}{D_{is}} \times F_{p1} \quad \text{(Eq. 3.63, Page 107)} \tag{12}$$

$$K_{st} = \sqrt{K_s^2 - K_{sm}^2} \quad \text{(Page 297)} \tag{13}$$

Separates the current into magnetizing and torque-producing components.[1]

10. **Finding No of Turns**

$$V_m = \frac{V_s}{\left(\frac{P_{ls}}{P_{ms}}\right)\left(1 + \frac{K_{st}}{K_{sm}}\right) + 1} \quad \text{(Page 298)} \tag{14}$$

$$\lambda_m = \frac{V_m}{2\pi f} \quad \text{(Pg.298,Step 13)} \tag{15}$$

$$N_s = \frac{\lambda_m P}{2k_1 B_{g1} l_e D_{is}} \quad \text{(Eq. 3.87, Page 298)} \tag{16}$$

Calculates Equivalent Magnetizing Voltage and Flux Linkage. Using above calculating the required turns per phase.[1]

3.2 Stator and Rotor Dimensional calculation

1. Input power calculation:

$$P_{in} = \frac{P_{out}}{efficiency} \tag{17}$$

2. Phase current calculation:

$$I_{ph} = \frac{P_{in}}{3 \times V_s \times PF} \tag{18}$$

3. Flux density and stack length:

   - $B_{ts} = B_t$ (maximum allowed flux density in the stator tooth)
   - $l_i = l_e$ [1]

4. Slot and conductor calculations:

   - Slots per phase:

$$slots\_per\_phase = \frac{stator\_slots}{3}$$

- Conductors per phase:

$$conductor\_per\_phase = 2 \times N_s$$

- Total conductors:

$$total\_conductors = num\_phases \times conductor\_per\_phase$$

- Total copper area:

$$A_{cu\_total} = total\_conductors \times \left(\frac{I_{ph}}{J_{s\_rms}}\right)$$

5. Slot-specific calculations:

   - Conductors per slot:

$$conductor\_per\_slot = \left(\frac{total\_conductors}{stator\_slots}\right)$$

   - Copper area per slot:

$$A_{cu\_per\_slot} = conductor\_per\_slot \times \left(\frac{I_{ph}}{J_{s\_rms}}\right) [1]$$

   - Slot area:

$$A_{slot} = \frac{A_{cu\_per\_slot}}{k_{fill}} [1]$$

6. Slot shape calculations:

   - For Trapezoidal slot:

$$\begin{aligned} s_{width\_avg} &= \sqrt{\frac{A_{slot}}{dim\_fatcor}} \\ s_{height} &= dim\_fatcor \times s_{width\_avg} \\ s_{width2} &= \left(\frac{2}{1 + dim\_factor1}\right) \times \sqrt{\frac{A_{slot}}{dim\_fatcor}} \\ s_{width1} &= dim\_factor1 \times s_{width2} \end{aligned}$$

   - For Rectangular slot:

$$\begin{aligned} s_{width1} &= \sqrt{\frac{A_{slot}}{dim\_fatcor}} \\ s_{height} &= dim\_fatcor \times s_{width1} \\ s_{width2} &= s_{width1} \end{aligned}$$

7. Stator dimensions:

$$\begin{aligned} d_{cs} &= \left(\frac{D_{os}}{2}\right) - \left(\frac{D_{is}}{2}\right) - s_{height} - tooth\_top\_height \\ \tau_s &= \pi \times 2 \times \left(\frac{D_{os}}{2} - d_{cs}\right) / stator\_slots[1] \\ t_{ts1} &= \tau_s - s_{width1} \\ t_{ts2} &= \tau_s - s_{width2} \\ t_{th} &= s_{height} + tooth\_top\_height \end{aligned}$$

8. Rotor dimensions:

$$D_{ir} = D_{is} - 2 \times g \quad \text{(rotor outer diameter)}$$
$$\tau_r = \pi \times D_{ir}/rotor\_slots \quad \text{(rotor slot pitch)}[1]$$
$$t_{tr} = \tau_r \times \left(\frac{B_{g1}}{B_{ts}}\right) \times \left(\frac{l_e}{l_i}\right) [1]$$
$$r_{width} = \tau_r - t_{tr}$$
$$r_{height} = \frac{s_{height}}{1.5}[4]$$

3.3 Magnetizing Inductance and Current

1. Magnetizing inductance calculation:

$$L_m = \frac{3\pi}{4} \cdot \left(\frac{4k_1N_s}{\pi P}\right)^2 \cdot \left(\frac{\mu_0 D_{is} l_e}{g_e}\right) \tag{19}$$

   Reference: Equation 3.85, Page 120[1]

2. Magnetizing current calculation:

$$I_m = \frac{V_s}{2\pi f L_m}[1] \tag{20}$$

   where V_s is the stator phase voltage and f is the supply frequency.

   Design Significance:

   - L_m determines the no-load current and power factor
   - Affects the motor's magnetizing VAR requirements
   - Critical for efficiency and performance calculations

3. **Verification Steps:**

   - Recalculated flux airgap density: $B_{g1,calc} = \frac{P\phi_p}{2D_{is}l_e}$ (Eq. 6.42)[1]

4 Code Architecture and Logic

4.1 Script Structure

The Lua script follows a modular top-down flow that mirrors the steps of classical machine design. It is structured into the following segments:

- **Custom Math Functions:** The script begins with user-defined functions like `sqrt()`, `pow()`, `sin()`, `round()`, `my_mod()` etc., allowing operation without relying on Lua's standard libraries. This improves portability and FEMM integration.

- **User Inputs Section:** All machine nameplate data, material types, slot shape preferences, and design constants like fill factor and flux density limits are provided at the top for clarity and configurability.

- **Electromagnetic Calculations:** This section computes torque, flux density, slot area, airgap, turns per phase, MMF, magnetizing inductance, and so on, mostly derived from Lipo's and Say's textbook equations.
- **Geometric Parameter Estimation:** From the electrical design, dimensions like stator/rotor slot width, height, tooth thickness, pitch, etc., are derived. Slot shapes are calculated based on chosen design constraints (`dim_factor`, `k_fill`, etc.).
- **FEMM Material and Problem Setup:** The `mi_addmaterial()` and `mi_probdef()` functions initialize FEMM's environment and materials (steel, copper, aluminum, air, etc.).
- **Drawing Logic:** Four different slot shapes (trapezoidal, rectangular, user-defined, and curved-corner versions) are implemented using geometry construction commands like `rotate()`, `mi_addnode()`, `mi_addsegment()`, and `mi_addarc()`.
- **Winding Layout Assignment:** The code assigns phase names and number of turns based on electrical slot angle using integer logic and custom `my_mod()` functions.
- **Labeling and FEMM Circuit Properties:** Block labels and circuit names (A, B, C, A1, etc.) are attached to the appropriate slots with directional angle and winding polarity control.

4.2 Key Functions

This section provides an overview of key custom functions used in the script. The functions are grouped into two categories:

4.2.1 Math Functions

To maintain portability and FEMM compatibility, all math operations like square root, power, sine, etc., are implemented using custom Taylor-series-based or simplified versions.

- `sqrt(x)`
  Computes the square root using the Babylonian method (also called Heron's method).

  ```
  function sqrt(x)
      local guess = x / 2
      for i = 1, 10 do
          guess = (guess + x / guess) / 2
      end
      return guess
  end
  ```

- `pow(x, y)`
  Calculates x^y using loop-based multiplication (supports only integer y).

```
function pow(x, y)
    if y == 0 then
        return 1
    end
    local result = 1
    local neg = false
    if y < 0 then
        y = -y
        neg = true
    end
    for i = 1, y do
        result = result * x
    end
    if neg then
        result = 1 / result
    end
    return result
end
```

- `rad(deg)`
  Converts degrees to radians.

```
function rad(deg)
    return deg * 3.1415926535898 / 180
end
```

- `Trigonometric functions`

```
function sin(x)
    local term = x
    local sum = x
    for i = 1, 10 do
        term = -term * x * x / ((2 * i) * (2 * i + 1))
        sum = sum + term
    end
    return sum
end

function cos(x)
    local term = 1
    local sum = 1
    for i = 1, 10 do
        term = -term * x * x / ((2 * i - 1) * (2 * i))
        sum = sum + term
    end
    return sum
end
```

- `abs(x)`
  Returns the absolute value of a number.

```
function abs(x)
    if x < 0 then
        return -x
    else
        return x
    end
end
```

- `truncate(x)`
  Removes the decimal portion of a number, similar to the integer truncation behavior (toward zero).

```
function truncate(x)
    local n = 0
    if x >= 0 then
        while n + 1 <= x do
            n = n + 1
        end
        return n
    else
        while n - 1 >= x do
            n = n - 1
        end
        return n
    end
end
```

- `round(x)`
  Rounds a number to the nearest integer, with correct handling for both positive and negative values. It uses your custom truncate() function internally.

```
function round(x)
    local int_part = truncate(x)
    local frac = x - int_part

    if x >= 0 then
        if frac >= 0.0 then
            return int_part + 1
        else
            return int_part
        end
    else
        if frac <= -0.5 then
            return int_part - 1
        else
            return int_part
        end
    end
end
```

- `my div(a, b)`
  Performs integer division of a by b. It returns the quotient and ignores the remainder.

```
function my_div(a, b)
  local count = 0
  while a >= b do
    a = a - b
    count = count + 1
  end
  return count
end
```

- `function optimal rotor slot(Ss, P)`
  Selects a suitable rotor slot number (S_r) based on the number of stator slots (S_s) and poles (P), ensuring it avoids known slot harmonic issues.

```
function optimal_rotor_slot(Ss, P)
    local Sr = Ss - 1

    while Sr > 0 do
        local diff = abs(Ss - Sr)

        if Sr ~= Ss and
           diff ~= P and
           diff ~= 2*P and
           diff ~= 3*P and
           diff ~= 5*P then
            return Sr
        end

        Sr = Sr - 1
    end

    return -1  -- No valid rotor slot found
end
```

4.2.2 Design Functions

- `rotate(x, y, d)`
  Rotates a point (x, y) by angle d (in radians) around the origin. This is essential for placing repeated geometry (e.g., stator/rotor slots) at equal angular intervals and it is used to rotate the slots and circuits.

```
function rotate(x, y, d)
    local xr = x * cos(d) - y * sin(d)
    local yr = x * sin(d) + y * cos(d)
    return xr, yr
end
```

- `upward offset(angle, tha))`
  Computes the point offset from the origin by a radial distance tha at a specified angular position angle (in radians). Internally uses the rotate() function.

  ```
  function upward_offset(angle,tha)
      local dx, dy = rotate(0, tha, angle)
      return dx, dy
  end
  ```

4.3 Parameter Handling

The script uses a clear and modular parameter handling structure to ensure flexibility and accuracy throughout the design.

4.3.1 User Inputs

At the top, the script collects all required inputs:

- Electrical specs: power, voltage, frequency, poles, slip, efficiency, and power factor.
- Material selections: for stator, rotor, winding, and shaft.
- Slot configurations: shape, type, number of slots.
- Design constants: fill factor, current density, winding factor.
- Slot ratios: dim_factor and dim_factor1 for height-to-width control.

4.3.2 Derived Parameters

Based on the inputs, the script computes:

- Airgap diameter and core length
- MMF, flux per pole, linear current density
- Turns per phase, slot dimensions, and winding area
- Rotor slot number using the optimal_rotor_slot() function

4.3.3 Geometry Handling

Slot dimensions are calculated conditionally based on the selected shape. The logic adjusts width, height, and tooth thickness using realistic ratios for thermal and magnetic viability.

This layered parameter structure ensures that the design process remains adaptable, readable, and simulation-ready.

5 Machine Design explanation

The stator and rotor are designed using parametric geometry with options enabling efficient flux flow and winding space. Both components are drawn in polar coordinates with customizable dimensions and replicated using angular steps to form a complete symmetrical machine.

5.1 Stator Design

5.1.1 General Design Logic

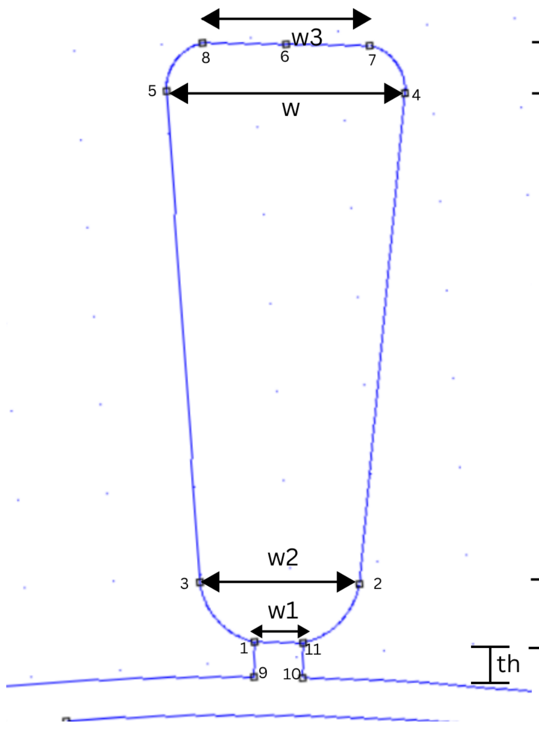


Fig. 1: Stator Slot Design

- **Slot Geometry Setup:** The code starts by defining a `draw_slot` function that calculates and draws a stator slot based on parameters:
  - Total height h
  - Bottom width w_1
  - Neck width w_2
  - Top arc width w_3
  - Body width w
  - Tooth top width th (all shown in the image)
- **Height Breakdown:** The total slot height h is split into sections:
  - h_1 for the bottom arc region

- h_3 for the top arc region
- $h_2 = h - h_1 - h_3$ for the central body

(These correspond to your right-side height labels)

- **Coordinate Rotation:** All (x, y) coordinates are computed using the `rotate()` function to place the slot at the correct angular position in the motor periphery, using polar conversion logic.
- **Nodes and Segments:** Key points (1 to 8 in image) are added as nodes to:
  - Draw straight lines and arcs
  - Define the trapezoidal body with curved corners using `mi_addsegment` and `mi_addarc`
- **Slot Opening Type:** If the slot is ”open” or ”semi_closed”:
  - Vertical lines (from points 1, 11 to 9, 10) are added to represent tooth tips
  - Uses `upward_offset()` logic and dynamic thickness control
- **Slot Replication:** The `draw_slot()` function is called in a loop for all slots based on the total number of stator slots. Each slot is placed using the calculated angle step (360/`stator_num_slots`).
- **Tooth Top Arcs:** For ”open” or ”semi_closed” slots, arcs are drawn between adjacent tooth tips (points 9 to next 10) to complete the outer closure of the stator teeth.
- **Stator Core Drawing:** If the slot type is ”closed”:
  - Only the inner and outer stator arcs are drawn
  - The stator outer circle is drawn in all cases using `mi_drawarc`

5.1.2 Trapezoidal with Curved Corners

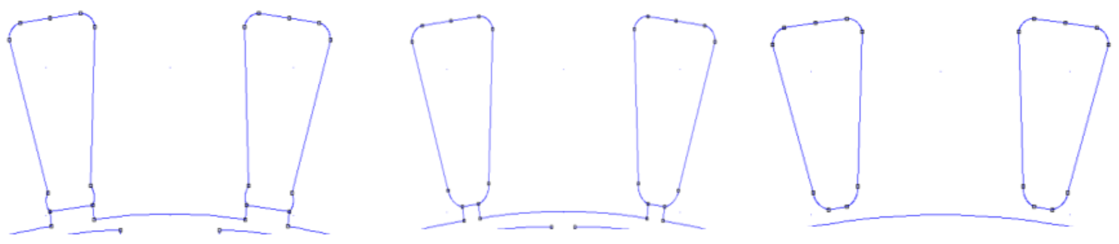


Fig. 2: Open, Semi closed, Closed Slot

- Combines a tapered trapezoid body and rounded top and neck
- Rounded corners are drawn using arcs
- Curved tooth tops are added for each slot with continuity

5.1.3 Rectangular with Curved Corners

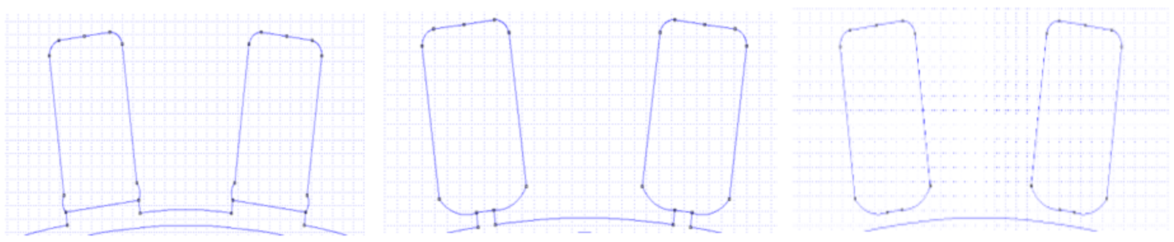


Fig. 3: open, semi closed, closed slot

- A uniform rectangular slot body with curved top and neck edges
- Slightly rounded tip improves mechanical strength and reduces saturation

5.1.4 Trapezoidal Slot

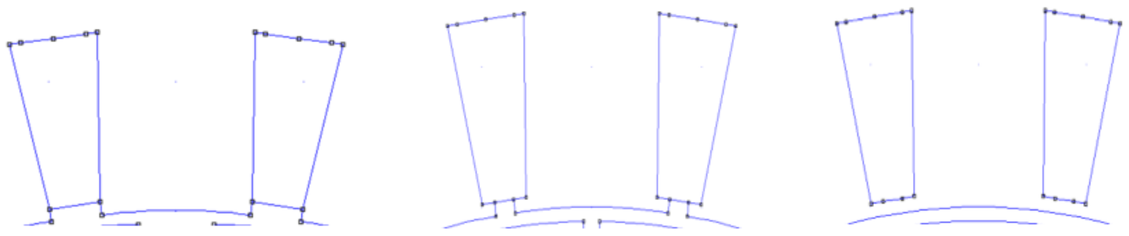


Fig. 4: open, semi closed, closed slot

- Straight trapezoidal body
- No curves; all segments are linear
- Arcs are replaced by sharp transitions

5.1.5 Rectangular Slot

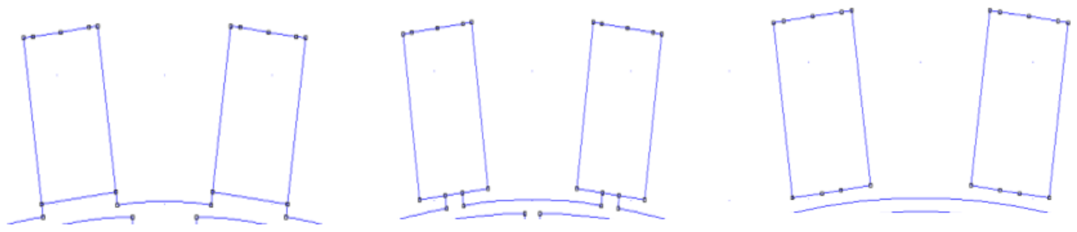


Fig. 5: open, semi closed, closed slot

- The simplest slot profile, entirely rectangular
- Used for comparison or where manufacturing simplicity is preferred

5.1.6 Designer-Defined Slot

- The designer manually enters height and width values for various segments
- Offers full customization for special applications or testing unusual designs
- Otherwise, Designer can use the separate design file also.In that file,Rectangular is the result, when the width r1 and r2 are same. Trapezoidal is the result, when the width r1 and r2 are different.

5.2 Rotor Design

The rotor design is similar to the stator in terms of angular replication and polar coordinate drawing. Each slot shape is generated using a `draw_slot()` function. Repeated around the shaft using `rotate()` and `upward_offset()` logic.The script supports five rotor slot shapes. Flexibility for open, semi-closed, and closed types

5.2.1 General Design Logic

- **Slot Geometry Construction:** Rotor slots are designed using parameters:
  - Total height h
  - Neck width r_2
  - Body width r_1
  - Top arc width r_3

  These define the overall keyhole or rectangular slot shape.
- **Height Segmentation:** Slot height h is divided into:
  - h_1 (neck height)
  - h_2 (arc region)
  - Remaining body height

  Vertically stacked from reference base `rotor_inner_slot_radius`.
- **Slot Positioning with Angle:** `rotate(x, y, angle)` places each slot at angular positions around rotor periphery, calculated using `rotor_angle_step = 360 / rotor_num_slots`.
- **Node Definitions:** Points x_1–x_8 define slot boundaries:
  - Base, neck, body, and top arc edges
  - Added with `mi_addnode(x, y)`
- **Boundary Connections:** Slot sides connected with:

    - `mi_addsegment(xA, yA, xB, yB)` for straight edges
    - `mi_addarc(xA, yA, xB, yB, angle, 1)` for curved regions

- **Slot Type Variations:** For "semi_closed" or "open" slots:
    - Vertical tooth tips added using offset points x_9, x_{10}
    - Computed via `upward_offset(angle, th)`
    - th is tooth tip height
- **Slot Replication:** `draw_slot()` called in loop from $i = 0$ to `rotor_num_slots - 1`, using `angle = i * rotor_angle_step`.
- **Tooth Top Arc Closure:** For open/semi-closed slots:
    - Arcs drawn between adjacent tooth tips
    - Using `connect_slot_bottoms()`
    - Calculates top positions based on `w_body`, th, and `rotor_inner_slot_radius + h`
- **Rotor Outer or Closed Boundary:** For "closed" slot type:
    - Continuous arc drawn on outer periphery
    - Using `mi_addarc(rotor_outer_radius, 0, -rotor_outer_radius, 0, 180, 1)`
- **Shaft and Core Design:**
    - Shaft modeled as circle with radius `shaft_radius = rotor_inner_slot_radius / 3`
    - Material regions defined with `mi_addblocklabel()`
    - Assigned using `mi_setblockprop()` with `rotor_core_material` and `shaft_material`

5.2.2 Trapezial Slot

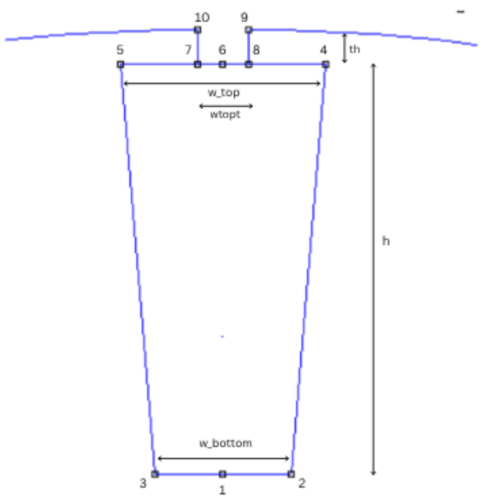


Fig. 6: Rotor Trapezoidal slot

- A trapezoidal body with or without an arc at the top
- Top and bottom widths differ: w_{top}, w_{bottom}
- Arcs are used for semi-closed shapes
- Arcs at tooth bottoms are connected slot-to-slot for continuity

5.2.3 Rectangular Slot

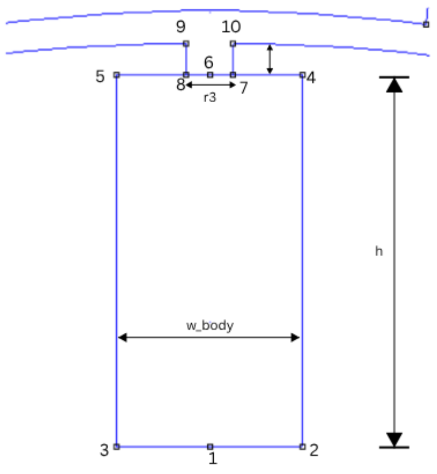


Fig. 7: Rotor Rectangular slot

- Simplest shape with constant width
- Straight vertical walls and flat base
- Suitable for simulations and analytical benchmarking
- Works with open/semi-closed variations by controlling arc tips

5.2.4 Circular Slot

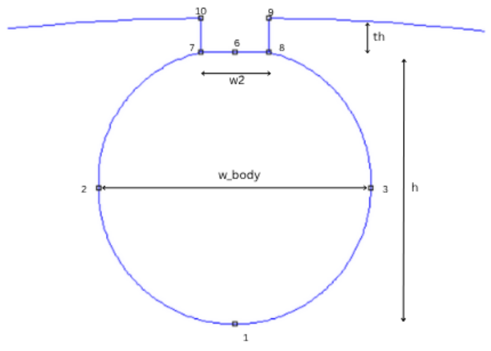


Fig. 8: Rotor Circular slot

- Slot is mostly a semi-circle

- Rounded body for reduced leakage inductance
- Suitable for squirrel cage rotors
- Arc angles vary based on slot-to-pitch ratio

5.2.5 Key Hole Single Cage Slot

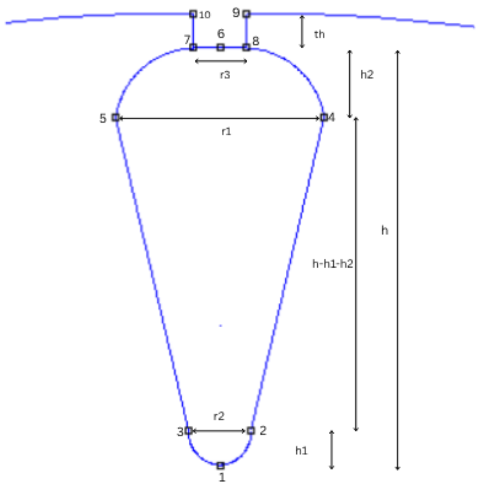


Fig .9: Rotor Keyhole Single cage slot

- Two-stage shape with:
  - Narrow neck and a circular cavity
  - Smooth transition using arcs
- Excellent for flux concentration and cooling

5.2.6 Key Hole Double Cage Slot

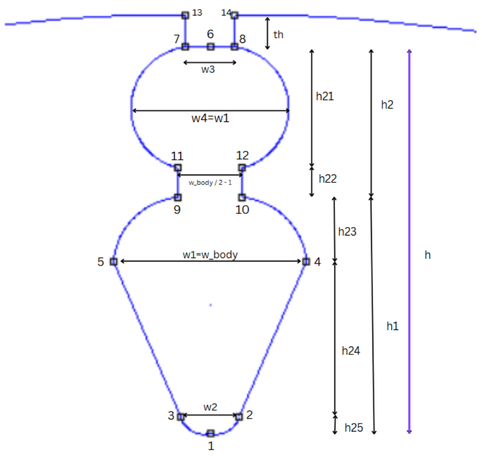


Fig. 10: Rotor Keyhole double cage slot

- Advanced version with:

  - Lower cavity (h_1), middle bridge (h_{22})
  - Upper cavity (h_{23}), and arc roof (h_{25})
  - Multiple arc transitions and segments
- Allows high starting torque in double cage induction motors

6 Winding Configuration

6.1 Stator Single layer Winding logic:

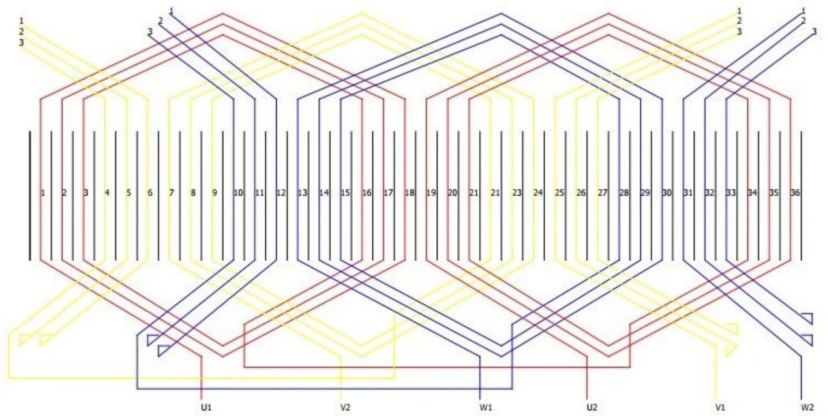


Fig. 11: Single Layer Winding for 36 Slots and 2 Pole Motor

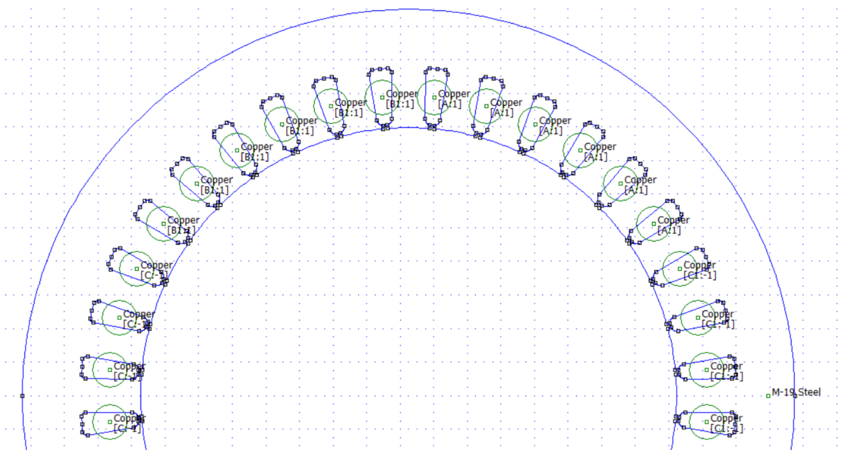


Fig. 12: Single Layer Winding for 36 Slots and 2 Pole Motor

- In a full-pitch winding:
  - The coil spans 180° electrical
  - Coil sides are placed exactly one pole apart
  - Maximizes induced EMF by aligning both coil sides with opposite magnetic polarities
- In a single-layer winding:

  - Each slot accommodates only one coil side (start or end)
  - Every slot is assigned a unique phase-turn value
  - Simplifies coil placement
  - Requires careful slot-phase assignment to preserve 120° phase displacement in 3-phase systems

6.2 Winding Code Logic:

- Slots per Pole per Phase (q):

  ```
  q = my_div(stator_num_slots, num_poles * num_phases)
  ```

  - This value is the base for deciding how slots are grouped for phase assignment
  - Example: with 48 slots, 8 poles, and 3 phases → q = 2

- Winding Phase Assignment

  ```
  block = my_div(i, q)
  phase_index = my_mod(block, 6)
  ```

  - Each slot index i is mapped to a repeating 6-slot cycle
  - The cycle assigns coils to phases: A, B, C, A1, B1, C1
  - A1, B1, C1 are return paths with reversed polarity

- Turns and Direction

  ```
  turns = round(turn_per_coil / 2)
  ```

  - Each slot gets half a coil (single layer)
  - Negative sign (turns = -turns) is used to reverse current direction based on phase, e.g., C and C1

- Phase Label Assignment

  ```
  if phase_index == 5 then phase_name = "A"
  elseif phase_index == 3 then phase_name = "B"
  elseif phase_index == 1 then phase_name = "C"
  elseif phase_index == 2 then phase_name = "A1"
  elseif phase_index == 0 then phase_name = "B1"
  elseif phase_index == 4 then phase_name = "C1"
  ```

  - Phase names are assigned using phase_index, maintaining 120° spacing

- Block Label Placement

  ```
  mi_addblocklabel(x, y)
  mi_setblockprop(stator_winding_material, 0, 1, phase_name,
                                  angle, 0, turns)
  ```

  - Labels are placed at each slot center (rotated polar coordinates)
  - Phase name, angle (0/±120), and number of turns are attached

6.3 Rotor Circuit Configuration

6.3.1 Cage Type Rotor

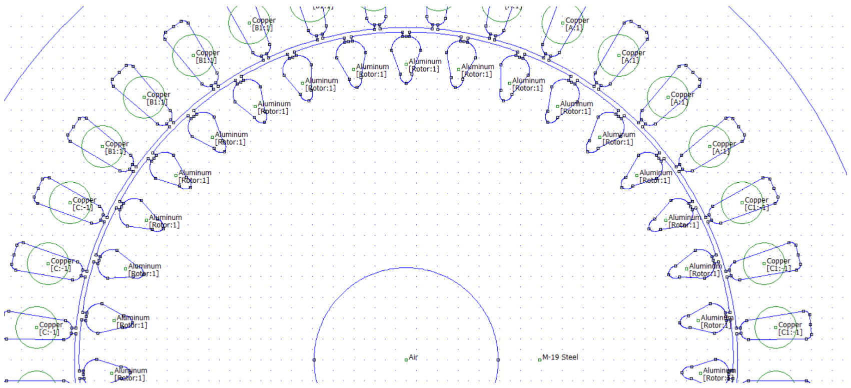


Fig. 13: Aluminium placed in the slots for 2 Pole Motor

- In a squirrel cage rotor:
  - Slots are filled with conductive bars (copper/aluminium)
  - Short-circuited at both ends by end rings to form closed loop
  - No need for external circuit connections
  - Stator field rotation induces current in rotor bars
  - Bars form parallel paths for induced current
  - Generates torque through Lorentz force
- Lua implementation:

```
if rotor_type == "cage_type" then
    mi_setblockprop(rotor_circuit_material, 1, 0, "Rotor",0, 2, 1)
```

  - Single label ”Rotor” added to each slot
  - No phase logic (all bars shorted via end rings)
  - `mi_addcircprop("Rotor", 0, 1)` treats cage as single passive conductor
- Visual reminder: Bars visible in slots, no terminals at ends

6.3.2 Wound Rotor Type

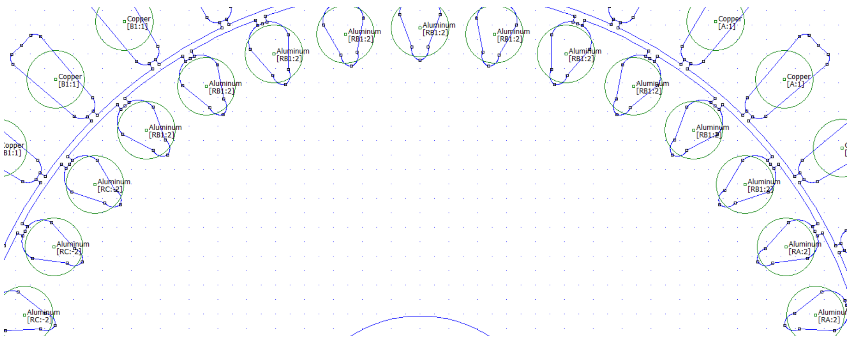


Fig. 14: rotor circuit placed in the slots for 2 Pole Motor

- Features:
  - Has actual windings in rotor slots (similar to stator)
  - Coils distributed in 3 phases (RA, RB, RC)
  - Return ends (RA1, RB1, RC1) internally shorted (Y-connection)
  - Start ends (RA, RB, RC) brought out through slip rings
  - Enables external current control (resistance starters/control systems)
  - Improves torque and speed control
- Lua implementation:

```
mi_addcircprop("RA", current, 1)
mi_addcircprop("RB", current, 1)
mi_addcircprop("RC", current, 1)
mi_addcircprop("RA1", -current, 1)
mi_addcircprop("RB1", -current, 1)
mi_addcircprop("RC1", -current, 1)
```

  - Each slot assigned label with phase name and direction
  - Phase logic follows same method as stator (using phase_index)
- Visual reminder: Simulation shows labeled blocks (RA, RB, etc.) in rotor slots

6.4 Simplified Summary with Visual Aid

- **Slot Grouping:**
  - Stator divided into repeating groups of 6 slots
  - Assigned to 3 phases (A, B, C) and their returns (A1, B1, C1)
- **Coil Configuration:**
  - Each slot holds one coil side (single-layer winding)
  - Positive turns (+) for supply current
  - Negative turns (-) for return current
- **Full-Pitch Advantage:**
  - Coils span one pole pitch (180° electrical)
  - Ensures optimal magnetic field reinforcement
- **Simulation Setup:**
  - Current directions controlled by turn sign (+/-)
  - Phase angles (0°, ±120°) assigned during block properties
  - Enables accurate EM field simulation

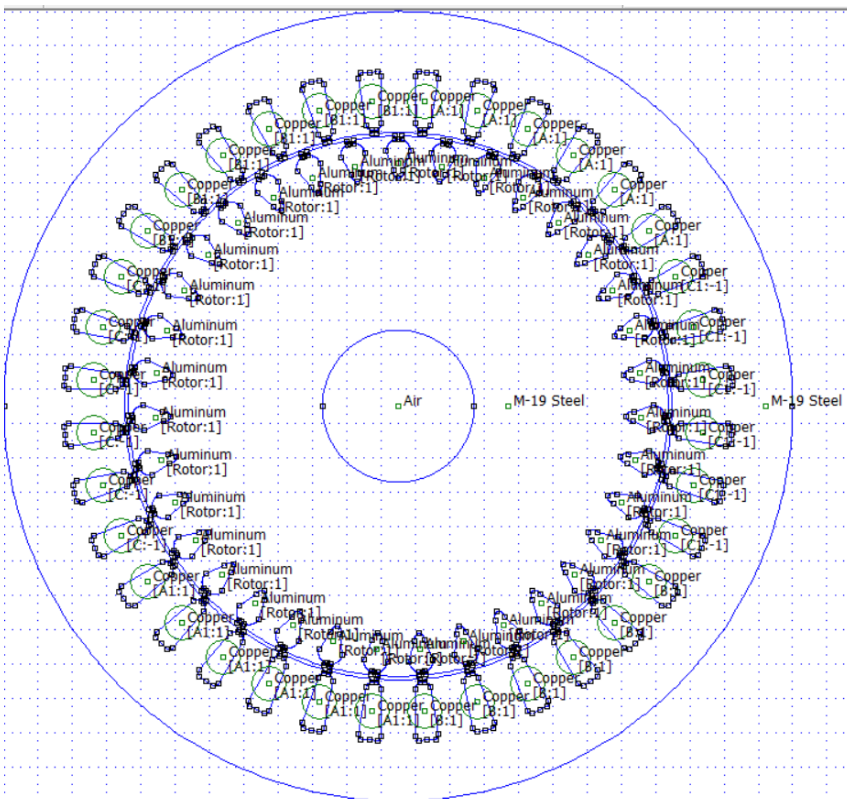


Fig. 15: 36 Slots 2 Pole Machine (Cage type Rotor)

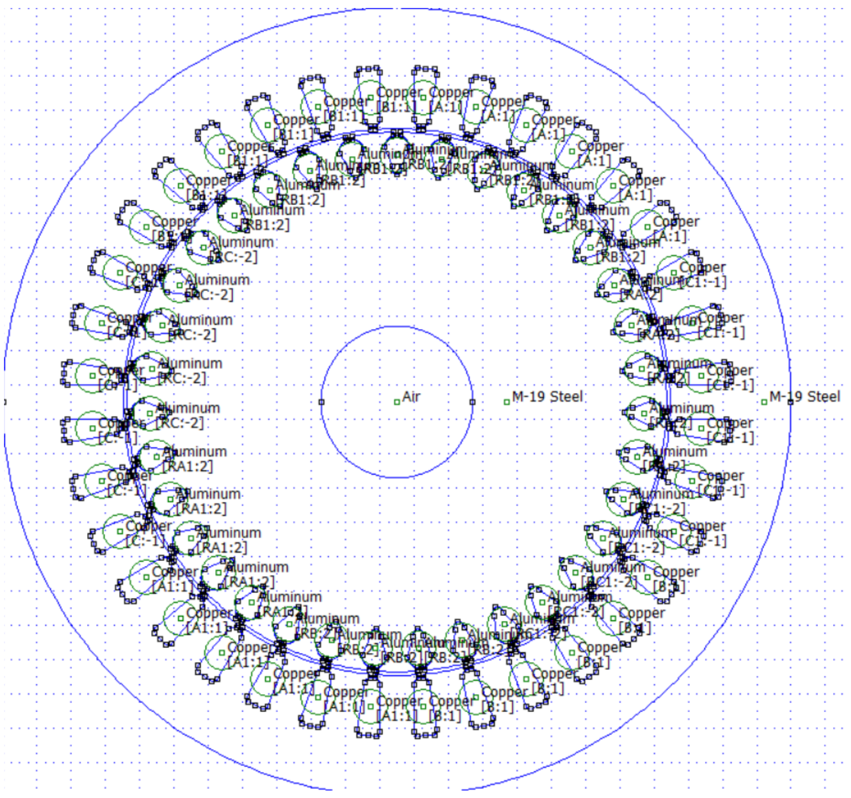


Fig. 16: 36 Slots 2 Pole Machine (wounded type Rotor)

7 FEMM Implementation

The design and analysis of the induction motor are performed using FEMM (Finite Element Method Magnetics) through an automated Lua script. The process consists of two main phases: Parameter Calculation and Model Design & Simulation.

7.1 Parameter Calculation

The script file `Rec_IM_single_file.lua` contains both the calculation and design parts for a complete induction motor.

- Start by providing your input values (such as power, voltage, speed, etc.) in the calculation section at the top of the Lua script.
- Copy this calculation section and run it in any online Lua compiler (e.g., repl.it or tutorialspoint) to view the computed design parameters.
- The output will display critical values like:
  - Outer stator diameter (Dos)
  - Inner stator diameter (Dis)
  - Axial length
  - Number of turns per phase
  - Slot height and widths
  - Magnetic loading ratios

7.2 Design Tuning Tips

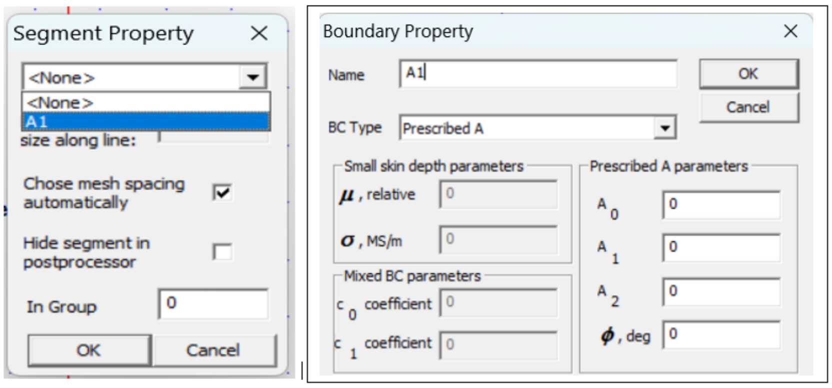
- If the Dos or axial length values seem too large or too small, adjust the input Dos and rerun the script.
- If the number of turns is odd or non-standard, try modifying `plm_pms_ratio` (pole loading ratio) between 0.2 and 0.4.
- If an error such as `negative value under square root in Dis/Dos calculation` appears, slightly adjust the a and b coefficients (related to stator/rotor ratio).
- Ensure that stator slot height is within realistic bounds. It should not exceed (Dos - Dis - core thickness).
- Verify the slot pitch, which should align well with the circumference $\pi\times$ (Dos - core thickness) to prevent overlap or crowding.

7.3 Geometry Creation and Simulation in FEMM

Once you're satisfied with the design parameters, follow these steps:

- Copy the entire script (including calculation and geometry drawing parts) into a .lua file.
- Open FEMM software, and load the .lua script using File → Open or directly drag and drop.
- The geometry will be automatically drawn including:
  - Stator core and slots
  - Rotor core and slots
  - Shaft
  - Airgap
  - Block labels and winding circuits (if wound rotor)

7.3.1 Boundary Conditions



size along line:
Chose mesh spacing automatically ✓
Hide segment in postprocessor ☐
In Group 0

Name A1
BC Type Prescribed A
Small skin depth parameters
μ, relative 0
σ, MS/m 0
Mixed BC parameters
c_0 coefficient 0
c_1 coefficient 0
Prescribed A parameters
A_0 0
A_1 0
A_2 0
ϕ, deg 0

Fig. 17: Segment and Boundary Properties

- Right-click each edge of the outer rectangle and press the space bar.
- Set the boundary condition to "A1". This confines the magnetic field within the simulation region.

7.3.2 Airgap Handling

- Zoom into the airgap area.
- Manually place the "Air" material in the region between the rotor and stator (if not already assigned).
- Ensure there is no contact between the rotor and stator to maintain a realistic magnetic airgap.

7.3.3 Final Steps

- Save the model as .fem.
- Go to the ”Mesh” menu and click ”Make Mesh” to prepare the domain for simulation.
- Run the simulation using ”Solve” and then ”View Results” in the Post Processor window.
- You can now view results such as:
  - Magnetic flux lines
  - Flux density distribution
  - Force, torque, and circuit quantities

8 RESULTS AND DISCUSSION

This section summarizes the output of the automated FEMM-based Lua design workflow for a representative input case: a 30 kW, 4-pole, 36-slot induction motor. The discussion evaluates key calculated values, performance insights, and how well the design aligns with theoretical expectations.

8.1 Sample Input Case (30kW, 400V, 2 Pole, 36 Slots)

8.1.1 Input Parameters:

Rating Inputs:

- Output Power: 30 kW
- Line Voltage: 400 V
- Frequency: 50 Hz
- Number of Poles: 2
- Stator Slots: 36
- Rotor Slots: Automatically selected (e.g., 35)
- Slip: 0.05
- Power Factor: 0.8
- Fill Factor: 0.6
- Efficiency : 0.95
- Number of Phases : 3

Winding and Density Input:

- Outer Stator diameter: 0.23 m

- RMS Current Density K_{rms}: 4 A/mm^2
- Winding Factor: 0.95
- Surface Current Density K_s: 8000 A/m
- Tooth Flux Density B_t: ~1.2 T (target)
- Airgap Flux Density B_{g1}: ~0.6 T (target)

Material Input:

- Stator Core Material: M-19 Steel
- Rotor core Material: M-19 Steel
- Stator Winding Material: Copper
- Rotor Circuit Material: Aluminium
- Shaft Material : Air

Slot Types and Dimensions:

- Stator Slot Shape : Trapezoidal curved corners
- Stator Slot Type : Semi Closed
- Rotor Slot Shape : Keyhole Single Cage Slot
- Rotor Slot Type : Semi Closed
- Rotor Type : Cage Type Rotor
- Dimension factor : 3 (Slot height/Slot width)(Stator)
- Dimension factor 1 : 1.5 (Slot width top/Slot width bottom)(For Trapezoidal Slots)

8.1.2 Manual Calculation :

- **Step 1: Synchronous and Mechanical Speed**

$$N_s = \frac{120 \times f}{P} = \frac{120 \times 50}{2} = 3000 \text{ rpm}$$

$$\omega_m = \frac{2\pi f}{(P/2)} = \frac{(2\pi \times 50)}{(2/2)} = 314.159 \text{ rad/sec}$$

- **Step 2: Electromagnetic Torque**

$$T_e = \frac{P_{out}}{\omega_m} = \frac{30000}{314.159} = 95.493 \text{ Nm}$$

- **Step 3: Air Gap Flux Density**

$$B_{g1} = 0.5 \times B_t = 0.5 \times 1.2 = 0.6 \text{ T}$$

- **Step 4: Linear Current Density**

$$K_s = 8000 \text{ A/m (Input)}$$

- **Step 5: Stator Inner Diameter Calculation (D_{is})**

$$a = 0.6, \quad b = 0.9$$

$$\frac{D_{is}}{D_{os}} = \left(\frac{b}{a} + \frac{2K_s}{ak_{cu}J_sD_{os}}\right) - \sqrt{\left(\frac{b}{a} + \frac{2K_s}{ak_{cu}J_sD_{os}}\right)^2 - \frac{1}{a}}$$

$$= \left(\frac{0.9}{0.6} + \frac{2 \times 8000}{0.6 \times 0.6 \times 4 \times 10^6 \times 0.23}\right) - \sqrt{\left(\frac{0.9}{0.6} + \frac{2 \times 8000}{0.6 \times 0.6 \times 4 \times 10^6 \times 0.23}\right)^2 - \frac{1}{0.6}}$$

$$= 0.69355m$$

- **Step 6: Feasibility Check of D_{is}/D_{os}**

$$\epsilon_{pci} = \frac{3a(D_{is}/D_{os})^2 - 4b(D_{is}/D_{os}) + 1}{((k_{cu}J_s)/4) \cdot [(D_{is}/D_{os})^2 - a]}$$

$$= \frac{3 \times 0.6 \times 0.6935^2 - 4 \times 0.9 \times 0.6935 + 1}{((0.6 \times 4 \times 10^6)/4) \cdot [(0.6935)^2 - 0.6]} = 8.8319 \times 10^{-6}$$

$$\epsilon_{pci} \geq 0 \quad \Rightarrow \quad D_{is} = 0.6935 \times 0.23 = 0.1595 \text{ m}$$

- **Step 7: Effective Length (l_e) Calculation**

$$a = 0.4, \quad b = 0.2$$

$$l_e = \frac{480 \cdot (P_{out}/\omega_{mech})}{\sqrt{2} \cdot \pi^2 \cdot k_1 \cdot k_{cu} \cdot \eta_{gap} \cdot \cos(\theta_{gap}) \cdot D_{os}^3 \cdot B_{g1} \cdot J_s \cdot (a(D_{is}/D_{os})^3 - 2b(D_{is}/D_{os})^2 + (D_{is}/D_{os}))}$$

$$= \frac{480 \cdot (30000/(2\pi \times 1500/60))}{\sqrt{2} \cdot \pi^2 \cdot 0.95 \cdot 0.6 \cdot 0.95 \cdot \cos(0) \cdot 0.23^3 \cdot 0.6 \cdot 4 \times 10^6 \cdot (0.4 \times 0.694^3 - 2 \times 0.2 \times (0.694)^2 + 0.694)}$$

$$= 0.3454 \text{ m}$$

$$\text{Aspect ratio} = \frac{P \times l_e}{\pi \times D_{is}} = \frac{2 \times 0.3454}{\pi \times 0.1595} = 1.3786$$

- **Step 8: Estimating MMF per Pole F_{p1}**

$$g = 0.003 \cdot \sqrt{P/2} \cdot \tau_p = 0.003 \cdot \sqrt{2/2} \cdot (\pi \times 0.23)/4 = 5.9968 \times 10^{-4} = 0.6 \text{ mm} \approx 1 \text{ mm}$$

$$g_e = K_c \cdot g = 2.5 \times 1 = 1.3 \text{ mm} \quad (K_c = 2.5)$$

$$F_{p1} = \frac{0.6 \times 0.0006}{\mu_0} = 1293.75 \text{ AT}$$

- **Step 9: Magnetizing Component of Surface Current Density**

$$K_{sm} = (P/D_{is}) \cdot F_{p1} = (2/0.1595) \times 1293.75 = 16222.57 A/m$$

- **Step 10: Torque-Producing Component of Surface Current**

$$K_{st} = \sqrt{K_{sm}^2 - K_s^2} = \sqrt{16222.57^2 - 8000^2} = 14112.823 Nm$$

- **Step 11: Permeance Ratio (P_{ls}/P_{ms})**

$$P_{ls}/P_{ms} = 0.25$$

- **Step 12: Equivalent Magnetizing Voltage V_m**

$$V_s/V_m = P_{ls}/P_{ms} \cdot (1 + K_{st}/K_{sm}) + 1 \Rightarrow V_m = V_s/\text{ratio}$$

$$V_s/V_m = 0.25 \cdot (1 + 14112.823/16222.57) + 1 = 1.4674$$

$$V_m = (400/\sqrt{3})/1.4674 = 157.38 \text{ V}$$

- **Step 13: Flux Linkage λ_m**

$$\lambda_m = V_m/2\pi f = 149.977/(2\pi \times 50) = 0.5 Wb$$

- **Step 14: Turns per Phase N_s**

$$N_s = \frac{\lambda_m \cdot P}{2 \cdot k_1 \cdot B_{g1} \cdot l_e \cdot D_{is}} = \frac{0.5 \times 2}{2 \times 0.95 \times 0.6 \times 0.3454 \times 0.1595} = 15.9225 \approx 16 \text{ turns}$$

- **Stator Slot Dimension:**
  - Input Power:

$$P_{\text{in}} = \frac{P_{\text{out}}}{\eta} = \frac{30{,}000}{0.95} = 31{,}578.94\,\text{W}$$

  - Phase Voltage:

$$V_{\text{ph}} = \frac{400}{\sqrt{3}} = 230.94\,\text{V}$$

  - Phase Current:

$$I_{\text{ph}} = \frac{P_{\text{in}}}{3 \cdot V_{\text{ph}} \cdot \text{PF}} = \frac{33{,}333.33}{3 \times 230.94 \times 0.8} = 56.97\,\text{A}$$

– Slots per phase:

$$slots\_per\_phase = \frac{stator\_slots}{3} = \frac{36}{3} = 12$$

– Conductors per phase:

$$conductor\_per\_phase = 2 \times N_s = 2 \times 16 = 32$$

– Total conductors:

$$total\_conductors = num\_phases \times conductor\_per\_phase = 3 \times 32 = 96$$

– Total copper area:

$$A_{cu\_total} = total\_conductors \times \left(\frac{I_{ph}}{J_{s\_rms}}\right) = 96 \times \left(\frac{56.97}{4 \times 10^6}\right) = 1367.28mm^2$$

– Conductors per slot:

$$conductor\_per\_slot = \left(\frac{total\_conductors}{stator\_slots}\right) = \left(\frac{96}{36}\right) = 2.667 \approx 3$$

– Copper area per slot:

$$A_{cu\_per\_slot} = 3 \times \left(\frac{56.97}{4 \times 10^6}\right) = 42.7275mm^2$$

– Slot area:

$$A_{slot} = \frac{A_{cu\_per\_slot}}{k_{fill}} = \frac{42.7275}{0.6} = 71.2125mm^2$$

– Slot Height and Width:

$$\begin{aligned} s_{height} &= 3 \times 4.8691 = 1.6807cm \\ s_{width2} &= \left(\frac{2}{1 + dim\_factor1}\right) \times \sqrt{\frac{A_{slot}}{dim\_fatcor}} \\ &= \left(\frac{2}{1+1.5}\right) \times \sqrt{\frac{71.2125}{3}} = 0.452cm \\ s_{width1} &= 1.5 \times 3.8976 = 0.675cm \end{aligned}$$

– Stator dimensions:

$$\begin{aligned} d_{cs} &= \left(\frac{23}{2}\right) - \left(\frac{15.95}{2}\right) - 1.6807 - 0.1 = 1.7443cm \\ \tau_s &= \pi \times 2 \times \left(\frac{23}{2} - 1.7443\right)/36 = 1.7026cm \\ t_{ts1} &= 1.7026 - 0.675 = 1.027cm \\ t_{ts2} &= 1.7026 - 0.452 = 1.2506cm \\ t_{th} &= s_{height} + tooth\_top\_height = 1.6807 + 0.1 = 1.7807cm \end{aligned}$$

- **Rotor Slot dimensions:**

$$\begin{aligned}
D_{ir} &= D_{is} - 2 \times g = 15.95 - 2 \times 0.1 = 15.75cm \\
\tau_r &= \pi \times D_{ir}/rotor\_slots = \pi \times 15.75/35 = 1.4137cm \\
t_{tr} &= \tau_r \times \left(\frac{B_{g1}}{B_{ts}}\right) \times \left(\frac{l_e}{l_i}\right) = 1.4137 \times \left(\frac{0.6}{1.2}\right) = 0.70685cm \\
r_{width} &= 1.4137 - 0.70685 = 0.70685cm \\
r_{height} &= \frac{1.6807}{1.5} = 1.1204cm
\end{aligned}$$

- **Magnetizing inductance calculation:**

$$L_m = \frac{3\pi}{4} \cdot \left(\frac{4 \times 8000 \times 16}{2\pi}\right)^2 \cdot \left(\frac{\mu_0 \times 0.1595 \times 0.3454}{2.5}\right) = 9.61mH$$

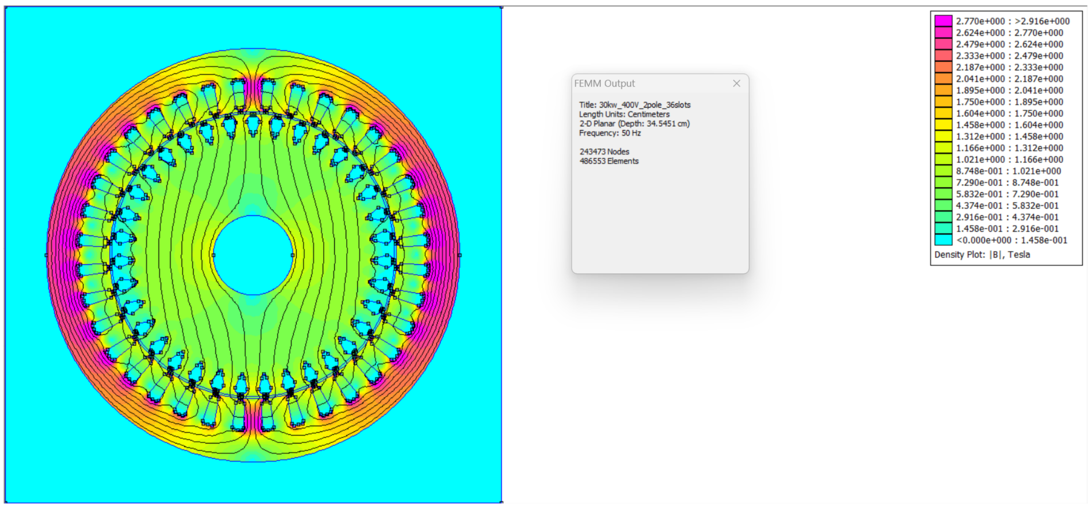
- **Magnetizing current calculation:**

$$I_m = \frac{V_s}{2\pi f L_m} = \frac{230.94}{2\pi \times 50 \times 0.00961} = 76.46A$$

- **Verification Steps:**
  - Recalculated flux airgap density: $B_{g1,calc} = \frac{P\phi_p}{2D_{is}l_e} = 0.6866T$

8.2 Design and Output Screenshots:

The FEMM simulation output shows a well-distributed and symmetrical flux pattern across the stator and rotor, confirming correct phase alignment and magnetic balance. A distinct two-pole flux pattern is observed, validating that the winding layout and pole pitch are accurately implemented. The flux lines smoothly bridge the air gap between stator and rotor, indicating effective electromagnetic coupling and minimal leakage. Additionally, the interaction of flux with the rotor bars confirms that the squirrel cage rotor is correctly energized and torque production is feasible under the given excitation.



```
FEMM Output

Title: 30kw_400V_2pole_36slots
Length Units: Centimeters
2-D Planar (Depth: 34.5451 cm)
Frequency: 50 Hz

243473 Nodes
486553 Elements
```

```
2.770e+000 : >2.916e+000
2.624e+000 : 2.770e+000
2.479e+000 : 2.624e+000
2.333e+000 : 2.479e+000
2.187e+000 : 2.333e+000
2.041e+000 : 2.187e+000
1.895e+000 : 2.041e+000
1.750e+000 : 1.895e+000
1.604e+000 : 1.750e+000
1.458e+000 : 1.604e+000
1.312e+000 : 1.458e+000
1.166e+000 : 1.312e+000
1.021e+000 : 1.166e+000
8.748e-001 : 1.021e+000
7.290e-001 : 8.748e-001
5.832e-001 : 7.290e-001
4.374e-001 : 5.832e-001
2.916e-001 : 4.374e-001
1.458e-001 : 2.916e-001
<0.000e+000 : 1.458e-001
Density Plot: |B|, Tesla
```

Fig. 18: Output and Flux Distribution of 30kW 400V 2 Pole 36 Slot Machine

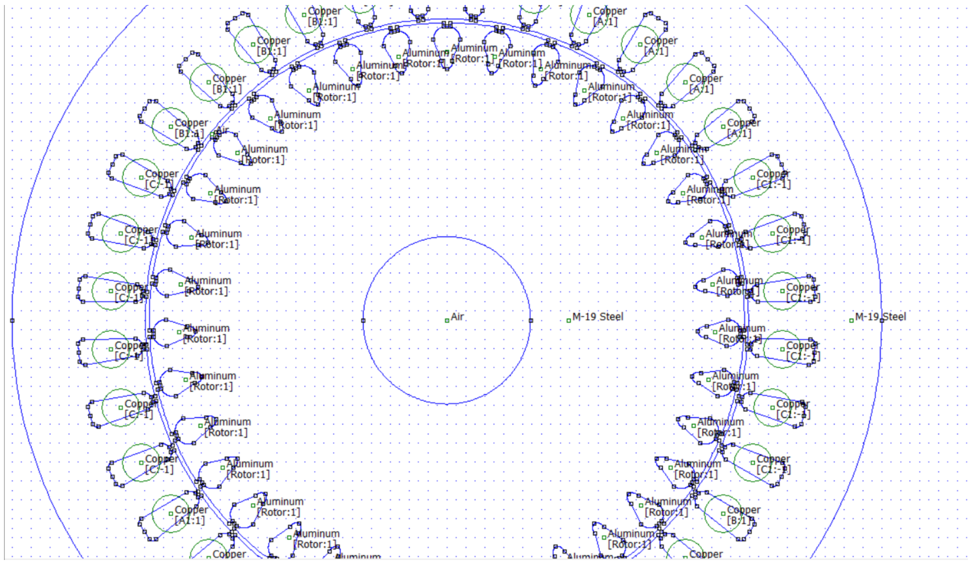


Fig. 19: Design of 30kW 400V 2 Pole 36 Slot Machine

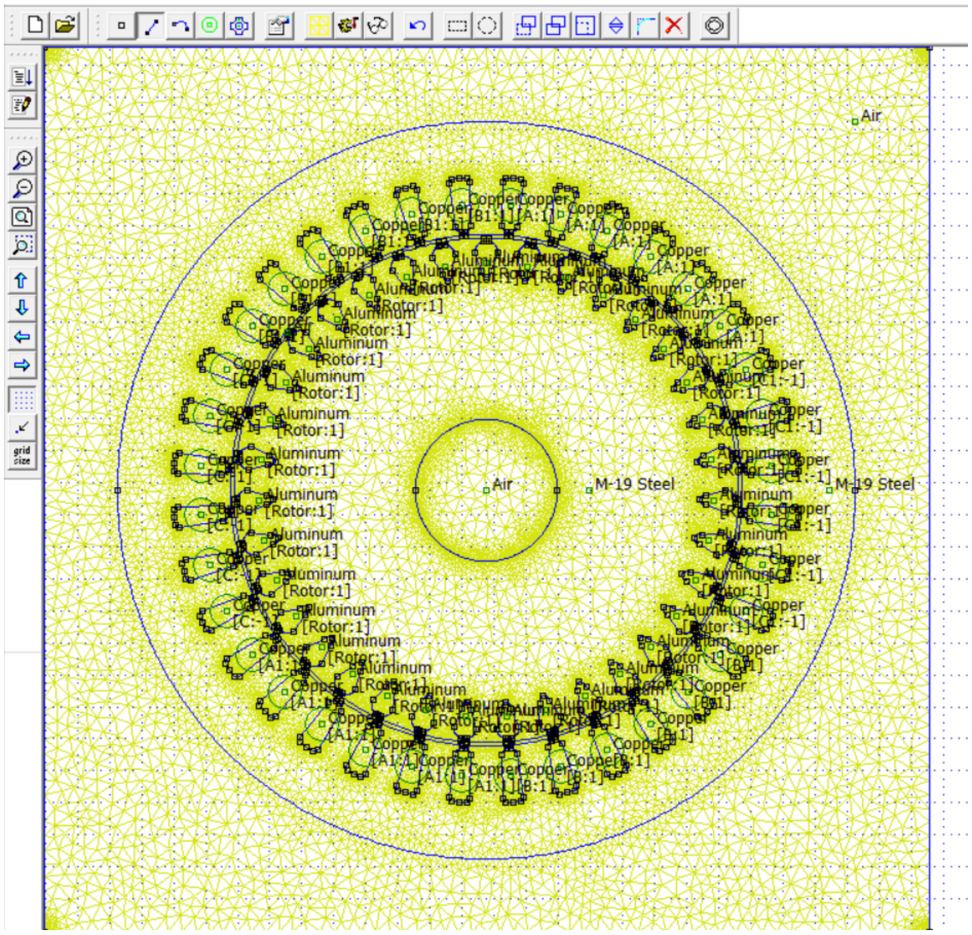


Fig. 20: Creating Mesh of 30kW 400V 2 Pole 36 Slot Machine

8.3 Flux Density Analysis:

8.3.1 Airgap Flux Density:

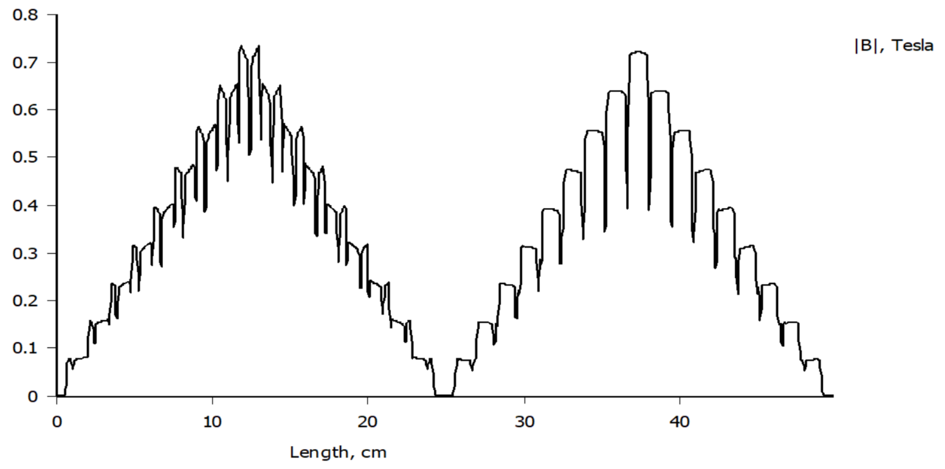


Fig. 21: Airgap Flux Density Plot

- The plot clearly shows two main peaks, confirming the 2-pole configuration of the machine and the correct implementation of the winding layout.
- The stator winding is designed (either distributed or with a specific coil span) to produce a rotating magnetic field. Ideally, this field varies sinusoidally in space along the air gap:

$$B(\theta) \propto \cos\left(\frac{P \cdot \theta}{2}\right) \tag{21}$$

- Moving along the air gap circumference:
  - Field strength gradually increases
  - Reaches peak at magnetic pole center
  - Decreases toward the inter-polar region
- The smooth envelope corresponds to the fundamental (first harmonic) of the air gap field produced by the main pole flux.
- Ripple patterns superimposed on the waveform are caused by:
  - Slotting effect (stator slots interrupting air gap uniformity)
  - Space harmonics from discrete coil placement in slots
- Despite harmonics, the waveform remains:
  - Symmetrical
  - Centered

indicating balanced magnetic operation and proper stator-rotor electromagnetic coupling.

8.3.2 Stator Flux Density:

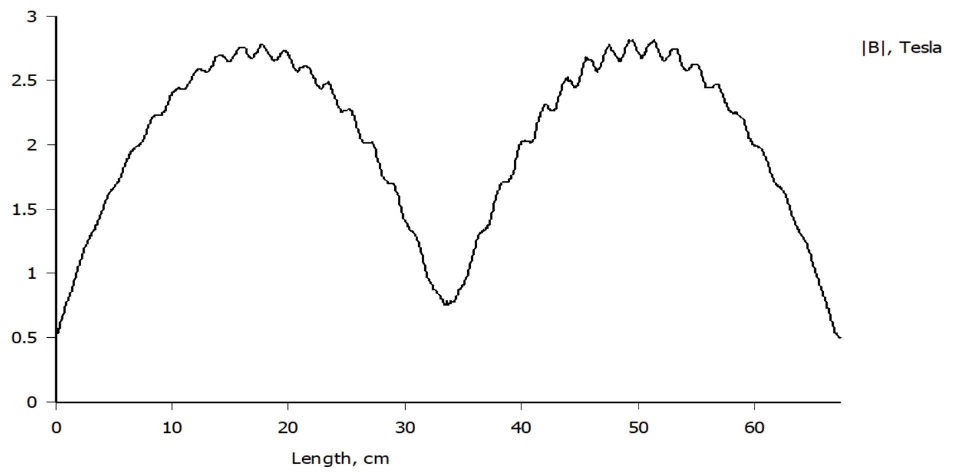


Fig. 22: Stator Flux Density Plot

- The waveform shows a symmetrical flux density pattern with two main peaks, consistent with the 2-pole excitation of the machine.
- The flux density gradually increases from both ends toward each pole center, forming a smooth envelope, indicating proper field buildup in the stator teeth and yoke.
- Small oscillations or ripples are observed across the peak region due to slot harmonics, resulting from the stator's discrete slot geometry.
- The waveform confirms that the stator is well magnetized, with magnetic field distribution consistent with sinusoidal MMF and efficient flux linkage through the core.

8.3.3 Rotor Flux Density:

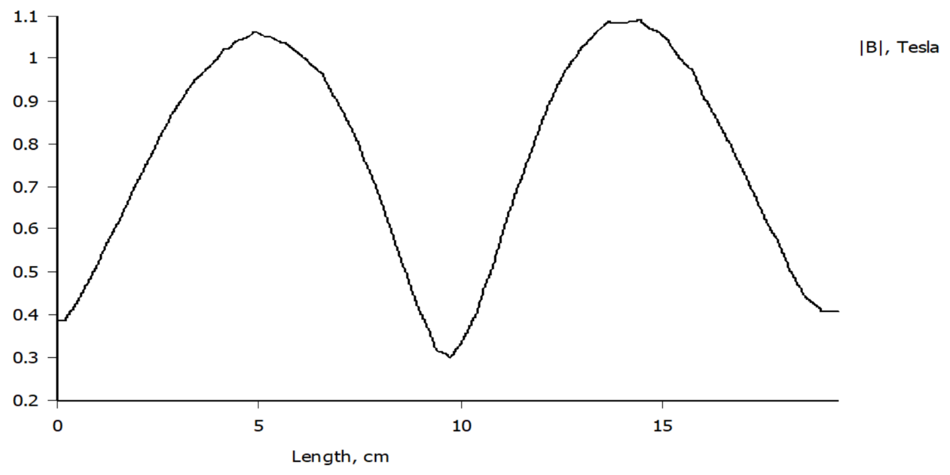


Fig. 23: Rotor Flux Density Plot

- The waveform displays two distinct flux density peaks, confirming correct 2-pole field induction in the rotor due to proper stator excitation and air-gap coupling.
- The smooth and symmetrical rise and fall of the flux density across the rotor surface reflects effective magnetic field penetration and uniform distribution across rotor teeth and yoke.
- The absence of significant high-frequency ripples indicates reduced harmonic content and smooth flux linkage in the rotor core, which is expected in cage-type rotors with uniform slot structure.
- The waveform confirms that the rotor is successfully reacting to the stator's rotating magnetic field, validating the overall electromagnetic design of the motor.

9 Conclusion

This project presents a generalized Lua-based script that integrates electromagnetic calculations and FEMM geometry modeling for induction motors. The script automates key steps such as input handling, dimension calculation, and drawing, significantly reducing design effort. It supports various stator slot geometries—rectangular, trapezoidal, and curved-corner types—offering flexibility for different design needs.

A key feature is the use of electrical angle-based winding logic, ensuring correct phase placement and pole formation across different slot-pole combinations. The simulation results, including flux plots and density waveforms, confirm smooth magnetic operation with sinusoidal MMF. Additionally, automated turn calculation and slot sizing based on power, voltage, and thermal limits make the tool practical and efficient for both study and early-stage motor prototyping.

9.1 Future Enhancements

- Add a graphical user interface (GUI) to improve accessibility for non-programmers and enhance usability for academic or industrial applications
- Extend support to double-layer and fractional-slot windings, enabling more compact and efficient motor designs
- Incorporate loss estimation modules (core loss, copper loss, stray load loss) and thermal analysis to evaluate efficiency and cooling requirements
- Implement multi-objective optimization routines for parameter tuning based on constraints like torque ripple, saturation, and material utilization
- Enable automatic report generation and export of CAD-ready geometries for manufacturing integration

10 Bibliography:

References

[1] Lipo, T. A. (2017). *Introduction to AC Machine Design.* WiscOnline Publishing.

[2] Say, M. G. (2005). *Alternating Current Machines* (5th Edition). CBS Publishers & Distributors.

[3] Say, M. G. *Performance and Design of AC Machines.* ELBS.

[4] Sawhney, A. K. *Electrical Machine Design.* Dhanpat Rai & Co.

[5] FEMM – Finite Element Method Magnetics, Version 4.2. `http://www.femm.info/wiki/HomePage`

[6] FEMM Lua Scripting Documentation. `http://www.femm.info/wiki/LuaScripting`

Design Files links:

1. The Full Single Design File With Calculations:
    - GitHub link.
    - Drive link.
2. The Design folder:
    - GitHub link.
    - Drive link.