UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELÉTRICA
CURSO DE ENGENHARIA ELÉTRICA

LUCAS EMANUEL BATISTUS FERREIRA

IMPLEMENTAÇÃO DE UM ALGORITMO PARA O CÁLCULO DO FATOR DE PREENCHIMENTO DE RANHURAS EM MÁQUINAS ELÉTRICAS SÍNCRONAS DE IMÃ PERMANENTE

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2019

LUCAS EMANUEL BATISTUS FERREIRA

IMPLEMENTAÇÃO DE UM ALGORITMO PARA O CÁLCULO DO FATOR DE PREENCHIMENTO DE RANHURAS EM MÁQUINAS ELÉTRICAS SÍNCRONAS DE IMÃ PERMANENTE

Trabalho de Conclusão do Curso de Engenharia Elétrica da Universidade Tecnológica Federal do Paraná - UTFPR, Câmpus Pato Branco, como requisito parcial para obtenção do título de Engenheiro Eletricista.

PATO BRANCO

2019

TERMO DE APROVAÇÃO

O Trabalho de Conclusão de Curso intitulado **IMPLEMENTAÇÃO DE UM ALGORITMO PARA O CÁLCULO DO FATOR DE PREENCHIMENTO DE RANHURAS EM MÁQUINAS ELÉTRICAS SÍNCRONAS DE IMÃ PERMANENTE** do acadêmico **Lucas Emanuel Batistus Ferreira** foi considerado **APROVADO** de acordo com a ata da banca examinadora **N°245** de **2019**.

Fizeram parte da banca examinadora os professores:

Prof. Dr. José Fabio Kolzer
Orientador

Prof. Msc. Cesar Augusto Portolann
Convidado 1

Prof. Dr. Edwin Choque Pillco
Convidado 2

A Ata de Defesa assinada encontra-se na Coordenação do Curso de Engenharia Elétrica

AGRADECIMENTOS

Este trabalho não seria possível sem a orientação do Professor José Fabio. Agradeço pelo auxílio e compreensão durante o desenvolvimento, e pela dedicação em transmitir sua experiência, em aulas e conversas externas.

Agradeço aos meus colegas que conheci e que me acompanharam durante o curso, Darlan, Pablo, Cristian e Eduardo e aos grandes amigos de longa data, Nicolas, Matheus, João, Jian, Abel, Otávio, Christian, Bayer e Thiago. Pelos momentos de descontração, conversas e pela ajuda durante esta jornada.

Agradeço ao meu pai Francisco, que sempre me apoiou incondicionalmente, e à minha mãe Ivania, por seus ensinamentos e pelo suporte. Sem o auxílio e a motivação de meus pais, não seria possível chegar aonde cheguei, nem ter a ambição de continuar.

A minha namorada Laura, seu carinho e sua dedicação me ajudaram a reencontrar a motivação necessária, que por muitas vezes pareceu sumir em meio a tantas preocupações.

RESUMO

FERREIRA, Lucas Emanuel Batistus. Implementação de um Algoritmo para o Cálculo do Fator de Preenchimento de Ranhuras em Máquinas Elétricas Síncronas de Imã Permanente. 2019. 75 f. Trabalho de Conclusão de Curso – Curso de Engenharia Elétrica, Universidade Tecnológica Federal do Paraná. Pato Branco, 2019.

A otimização de um projeto de máquinas elétricas que busque melhorar a eficiência na conversão entre energia mecânica e elétrica deve focar-se na atenuação do impacto causado pelas principais fontes de perdas nestes dispositivos, como as perdas Joule nos enrolamentos de cobre, correntes parasitas no núcleo ferromagnético, perdas mecânicas por atrito e ventilação, e outras perdas suplementares. O fator de preenchimento de ranhura é um dos parâmetros limitantes nessa otimização. Este trabalho foi desenvolvido com o intuito de implementar um algoritmo que realize o cálculo do maior fator de preenchimento de ranhura possível para uma determinada máquina, através do posicionamento ideal dos condutores dentro das ranhuras do estator, como consequência, obtendo uma redução na resistência térmica dos enrolamentos e uma diminuição na elevação de temperatura, o que causará também, um aumento na densidade de torque por volume.

Palavras-chave: Algoritmo. Fator de Preenchimento de Ranhura. Máquinas Elétricas.

ABSTRACT

FERREIRA, Lucas Emanuel Batistus. Implementation of an Algorithm for the Calculation of the Slot-Filling Factor in Permanent Magnet Synchronous Machines. 2019. 75 f. Trabalho de Conclusão de Curso – Curso de Engenharia Elétrica, Universidade Tecnológica Federal do Paraná. Pato Branco, 2019.

The optimization of an electrical machine project that seeks to improve the efficiency on the electrical to mechanical energy conversion should be focused mainly on the mitigation of the impact caused by the main sources of losses on this devices, such as the Joule losses at the winding, the eddy currents at the magnetic core, mechanical losses caused by friction and ventilation, and other additional losses. The slot filling factor is one of the limiting parameters on this optimization. This work was developed with the intent of implementing an algorithm that does the calculation of the maximum possible slot filling factor for a determined machine, through the ideal positioning of the wires inside the stator slots, and as consequence, achieving a decrease in the temperature rise, also increasing the torque density per volume.

Keywords: Algorithm. Slot-Filling Factor. Electrical Machines.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Tendo em vista a crescente busca pela automação e otimização de processos, a utilização de algoritmos computacionais é tida como uma ferramenta de grande importância neste contexto. Independente do processo a ser otimizado, resultados que ultrapassem os limites impostos pelos métodos usuais requerem uma análise muitas vezes iterativa, ou que demande de conceitos matemáticos severamente complexos.

Devido também a alta escalabilidade de alguns processos, torna-se algo intangível a realização manual dos métodos iterativos de otimização, o que ressalta novamente a importância do desenvolvimento e aplicação de algoritmos em soluções de diversas áreas.

O trabalho desenvolvido tem esse princípio como sua motivação raiz, a ideia de implementar um algoritmo para o cálculo do fator de preenchimento de ranhura descende da necessidade de definir um parâmetro que muitas vezes é tido como um dado de saída, um critério de natureza causal no projeto de máquinas elétricas. Citado brevemente em Hendershot e Miller (2010), este fator possui um papel importante nos projetos, com suas peculiaridades apresentadas e exploradas neste trabalho.

1.1 OBJETIVO GERAL

O objetivo principal deste trabalho consiste na implementação do algoritmo desenvolvido por Raabe (2014), com alguns ajustes para a aplicabilidade em outros formatos geométricos da ranhura de uma máquina elétrica, e em determinar um método para melhorar a obtenção prévia do fator de preenchimento de ranhura, o que ocasiona uma redução na resistência térmica do enrolamento de armadura da máquina em regime permanente, por meio da maximização do fator mencionado.

1.2 OBJETIVOS ESPECÍFICOS

1. Implementação do algoritmo para o cálculo do fator de preenchimento de ranhura.
2. Analisar a influência dos parâmetros geométricos da ranhura e dimensões dos condutores no fator de preenchimento da ranhura.
3. Relacionar as dimensões dos condutores com as dimensões da ranhura.
4. Realizar as análises para múltiplos tipos de ranhura.
5. Avaliar a importância dos parâmetros construtivos, como o número de polos da máquina, e de desempenho, como a corrente, nas dimensões da ranhura.

1.3 ORGANIZAÇÃO DO TRABALHO

A organização do conteúdo deste trabalho foi feita da seguinte forma. No capítulo 1 é descrita brevemente uma introdução ao tópico a ser discutido, com comentários relacionando a aplicação do conceito de algoritmos para automação de processos com o projeto e desenvolvimento de máquinas elétricas.

No capítulo 2 é apresentada a fundamentação teórica do trabalho, os conceitos e critérios que são empregados em projetos de máquinas elétricas, a definição das perdas nos enrolamentos e considerações sobre a influência do fator de preenchimento de ranhura.

Na sequência, o capítulo 3 descreve as ferramentas (*Python* e *Gmsh*) utilizadas e quais são suas peculiaridades, contido nesse capítulo está também, a apresentação dos métodos que foram empregados para a modelagem e representação matemática das ranhuras e dos condutores, bem como a descrição do algoritmo a ser implementado.

No capítulo 4 são descritos os resultados da implementação em cada uma das condições estabelecidas, posteriormente é feita uma análise comparativa entre duas ranhuras de máquinas de desempenhos idênticos. Após isso, comparam-se os resultados alcançados com os dos artigos referenciados e em seguida é apresentado o panorama geral dos dados obtidos.

No capítulo 5 são mostradas as conclusões que foram possíveis de serem retiradas do trabalho desenvolvido e são apresentadas sugestões para trabalhos futuros.

Como complemento ao tema abordado, são disponibilizados no Apêndice A os *scripts* do algoritmo implementado na linguagem *Python*. Cada módulo é explicado separadamente e é descrita a forma com que se relacionam para gerar os resultados obtidos. É apresentado também um exemplo de implementação, com as características de entrada a serem consideradas na execução do algoritmo.

2 REVISÃO DE LITERATURA

Este capítulo apresenta o embasamento teórico dos métodos e considerações utilizadas na implementação do algoritmo, bem como as justificativas para o trabalho desenvolvido e critérios a serem observados durante o projeto de uma máquina elétrica.

2.1 CRITÉRIOS DE PROJETO DE UMA MÁQUINA ELÉTRICA SÍNCRONA

Devido à extensa lista de aplicações nas que uma máquina síncrona pode ser utilizada, não existe um passo-a-passo geral para o projeto da mesma, torna-se então, imprescindível que sejam definidos previamente diversos fatores. Segundo Hendershot e Miller (2010), os principais fatores a serem definidos nos estágios iniciais de projeto de uma máquina são os seguintes:

1. Decidir a configuração da máquina e seu controle;
2. Especificar o número de fases, polos e ranhuras;
3. Estimar as principais dimensões e selecionar os materiais;
4. Projetar o rotor;
5. Projetar a laminação do estator e do pacote de chapas;
6. Projetar o enrolamento do estator.

Ao serem determinadas as condições de operação da máquina, são especificados os valores de potência, tensão e fator de potência nominal da mesma. Com esses dados, é possível obter a corrente nominal e então determinar o número de espiras do enrolamento e a seção condutora necessária para atender o projeto especificado.

O trabalho desenvolvido teve como foco principal o estudo de um dos parâmetros intrínsecos ao projeto da lâmina do estator e dos enrolamentos, presentes nos tópicos 5 e 6. Este parâmetro é denominado Fator de Preenchimento de Ranhura e sua obtenção é dada a partir da razão entre a seção condutora das espiras, considerando o verniz isolante nos fios, e a área útil da ranhura, determinada a partir da área total, subtraindo a área do pescoço da ranhura e dos isolantes de fechamento de ranhura e/ou entre fases (no caso de máquinas com enrolamento em camada dupla). Uma visualização algébrica deste fator é apresentada nas equações (1) e (2).

$$f_{ran} = \frac{N_f \cdot A_f}{A_{ran,util}} \tag{1}$$

Onde, N_f é o número de condutores, A_f é a seção de cada condutor, e $A_{ran,util}$ é a seção da ranhura.

A equação (1) utiliza a seção total dos condutores, considerando o isolamento que envolve cada um deles. Já a equação (2) leva em conta somente a seção de cobre, A_{cob}, sem considerar os isolantes, o que nos permite obter a relação entre a área total de cobre que está disposta na ranhura, visto que as perdas no cobre no estator são inversamente proporcionais à seção de cobre no mesmo, sendo então interessante maximizá-la para efeitos de desempenho (RAABE, 2014).

$$f_{cob} = \frac{N_f \cdot A_{cob}}{A_{ran}} \tag{2}$$

2.2 CONSIDERAÇÕES SOBRE O FATOR DE PREENCHIMENTO DE RANHURA

O fator de preenchimento de ranhura de uma máquina irá determinar o quão bem dispostos estão os condutores no interior da ranhura, ou seja, é relacionado com o quanto se está aproveitando de espaço dentro das ranhuras do estator, bem como na quantidade de ar que estará preenchendo o restante do meio, conforme pode ser observado na Figura 1.

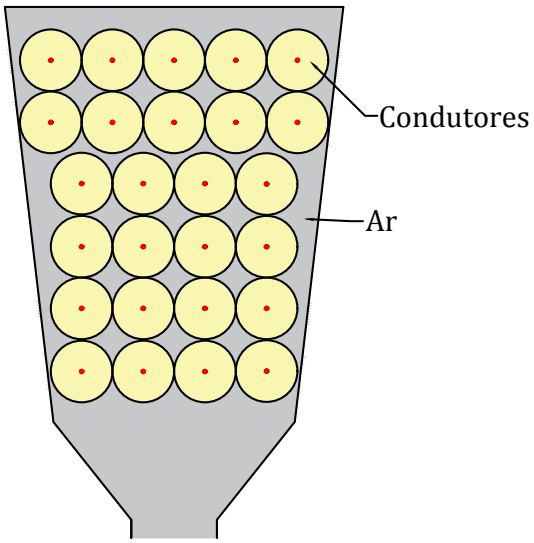


Figura 1 – Corte frontal da ranhura
Fonte: Autoria Própria

Dentro do projeto de uma máquina elétrica, o fator de preenchimento de ranhura é geralmente determinado a partir da estimativa manual, grosseira e inicial das dimensões do estator e então, são realizadas simulações, análises por elementos finitos, com as dimensões calculadas, posteriormente, ajustes e melhorias graduais são realizadas para uma melhor adequação ao projeto (KOLZER, 2017).

O processo manual de especificação e representação gráfica das dimensões da ranhura e disposição dos condutores em seu interior, além de ser um processo demorado, está sujeito a erros e resultados de níveis subótimos. Além disso, a confecção de um estator para testes iniciais gera custos e atrasos adicionais ao projeto (RAABE, 2014).

Um estudo realizado por Jack (2000), tem como objetivo a melhor utilização dos materiais para o projeto de uma máquina elétrica, sendo um de seus pontos de estudo o melhor aproveitamento térmico a partir de um aumento no fator de preenchimento da ranhura. Com esse estudo foi possível constatar, em seu artigo, que o aumento de 10% (partindo de um fator 0.7 para um de 0.8), foi possível alcançar uma redução de 46% na resistência térmica dos enrolamentos.

A correta especificação do fator de preenchimento de ranhura proverá melhorias pontuais de grande valia para o desempenho final da máquina, aumentando a densidade de torque por volume, por exemplo. Um valor reduzido de fator de preenchimento de ranhura é responsável, conforme comentado acima, por uma maior resistência térmica dos enrolamentos, aumento nas perdas do cobre, bem como uma pior redução da elevação de temperatura da máquina, devido à maior quantidade de ar presente internamente à ranhura (HENDERSHOT; MILLER, 2010; JAKSIC, 2011).

Devido ao efeito Joule, a condução de corrente elétrica pelos enrolamentos de cobre do estator é responsável pela maior parte das perdas de potência em máquinas elétricas. Estas perdas podem ser calculadas por $N_f I_f^2 R_f$, aonde N_f é o número de fases, I_f é a corrente RMS da fase e R_f a resistência. O aumento da temperatura nos condutores irá aumentar a resistência dos enrolamentos. Por exemplo, um aumento de 50 °C causa um incremento de 20% e 135 °C um acréscimo de 53% na resistência, o que irá causar um aumento nas perdas de forma proporcional, caso a corrente mantenha-se constante (HENDERSHOT; MILLER, 2010).

Segundo Asokan (ASOKAN, 2004 apud JAKSIC, 2011), há também a presença de efeito Corona nos condutores do enrolamento, diretamente associado com o fator de preenchimento e, consequentemente a seção dos condutores. Por outro lado, um valor extremamente elevado do fator de preenchimento irá dificultar na manufatura do dispositivo, podendo tornar-se inviável para a disposição dos condutores dentro da ranhura. Torna-se então interessante e de grande utilidade estudar e implementar um método para se determinar de forma rápida e eficaz o fator de preenchimento de ranhura.

2.3 CONSIDERAÇÕES

Com a premissa de automatizar e aumentar a confiabilidade do cálculo do fator de preenchimento de ranhura, a implementação de um algoritmo computacional é uma opção que disponibiliza uma forma de realizar os cálculos e adequações com agilidade e precisão.

O algoritmo desenvolvido por Raabe (2014) e posteriormente estudado por Tommaso (2017), é capaz de fornecer uma base no projeto das dimensões de ranhuras de diversos modelos geométricos, previamente parametrizados, e maximizar de forma viável o fator de preenchimento dos condutores dentro das ranhuras especificadas.

Com base nesses estudos, foi realizada a implementação do algoritmo descrito neste trabalho, que terá suas características aprofundadas nos próximos capítulos.

3 MATERIAIS E MÉTODOS

Neste capítulo serão descritos os conceitos utilizados na implementação do algoritmo, quais foram os parâmetros a serem considerados para a obtenção do modelo computacional dos condutores e das ranhuras, bem como a forma que será feita a interação entre estes dois objetos. Além disso, são apresentadas as ferramentas que tornaram possível esse desenvolvimento, e de que forma foram utilizadas.

3.1 CONSIDERAÇÕES INICIAIS

Para que fosse possível implementar uma sequência lógica que realizasse as ações desejadas foi necessário definir um ambiente de desenvolvimento e uma forma para que se pudesse obter uma visualização gráfica dos resultados.

De forma geral, a implementação do algoritmo consiste na interação entre formas geométricas, que precisam ser simuladas e ter seus contornos delimitados de forma precisa para que não haja sobreposição de objetos. Para isso, é necessária uma ferramenta que disponha de métodos para cálculos aplicados, análise, criação e manipulação de dados de listas e matrizes, bem como a possibilidade de percorrer e iterar sobre estes dados.

Uma ferramenta que possui todas essas características supracitadas é o *Python*, uma linguagem de programação, de alto nível, orientada a objetos e de propósito geral criada por *Guido van Rossum* e lançada em sua primeira versão em 1991 (MARTELLI, 2006).

Além de possuir diversas bibliotecas que permitem o desenvolvimento do trabalho, *Python* demanda muito pouco poder de processamento, podendo ser utilizado em máquinas de menor custo sem que haja uma redução considerável na performance. Outro ponto chave na decisão foi o fato de ser uma ferramenta de código aberto, que não necessita de licenças comerciais para sua utilização.

Para a visualização gráfica das ranhuras e os condutores, foi selecionado o aplicativo *Gmsh*[1], um *software* livre, de geração de malhas em elemento finitos e com uma *engine* de Desenho Assistido por Computador (do inglês *CAD*, *Computer Aided Design*) integrada. O intuito ao se escolher este programa foi a simplicidade e eficiência que o mesmo disponibiliza e a possibilidade de utilizar *scripts* para que os desenhos necessários possam ser obtidos. Na Figura 2 é possível visualizar a interface do *Gmsh*.

[1] http://gmsh.info/

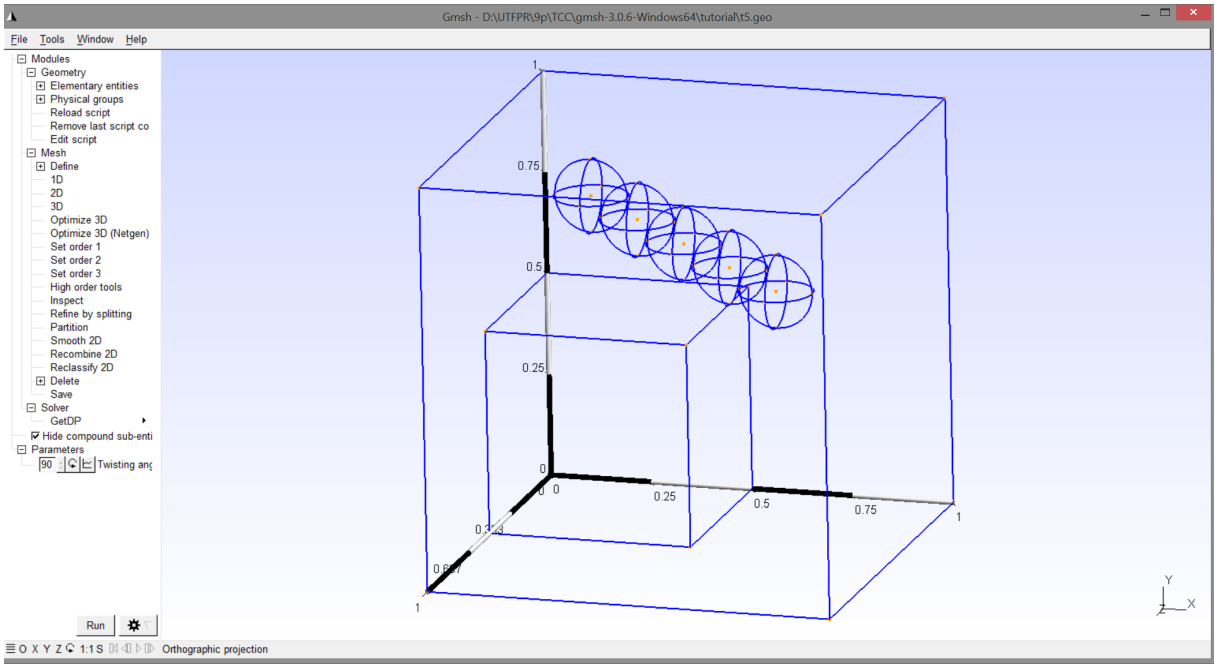


Figura 2 – *Gmsh*, *software* para geração dos desenhos
Fonte: Autoria Própria

3.1.1 *Python* e o Ambiente de Desenvolvimento Integral

Dividido em duas principais versões, *Python* 2.x e 3.x, optou-se por utilizar a última versão estável da segunda ramificação, 3.7, devido a versão 2 possuir algumas limitações e estar sendo descontinuada por parte dos desenvolvedores, ainda disponível apenas para *legacy support*.

Python, segundo Martelli (2006), "é uma linguagem simples, porém não simplista", e essa característica é ressaltada pela enorme gama de bibliotecas disponíveis, sejam elas integradas ou desenvolvidas pela comunidade. As bibliotecas utilizadas foram:

- *Numpy*[2]: Biblioteca que possui a implementação de diversas funções e constantes matemáticas, além de métodos para manipulação de listas e vetores.
- *Pandas*[3]: Biblioteca com diversas funções para manipulação de matrizes e dados em formato tabular.

Para aperfeiçoar a utilização da linguagem, foi selecionado um Ambiente de Desenvolvimento Integral (do inglês *IDE*, *Integral Development Environment*) chamado *Spyder*[4], que possui uma interface gráfica e um console para rodar o código enquanto se edita o *script* na mesma tela, o que possibilita uma interação mais assistida e com mais recursos. Na Figura 3 é apresentada a janela da IDE, sendo possível visualizar

[2] numpy.org/
[3] pandas.pydata.org/
[4] spyder-ide.org/

a seção de edição do *script* a esquerda, e na direita um explorador de variáveis e o console.

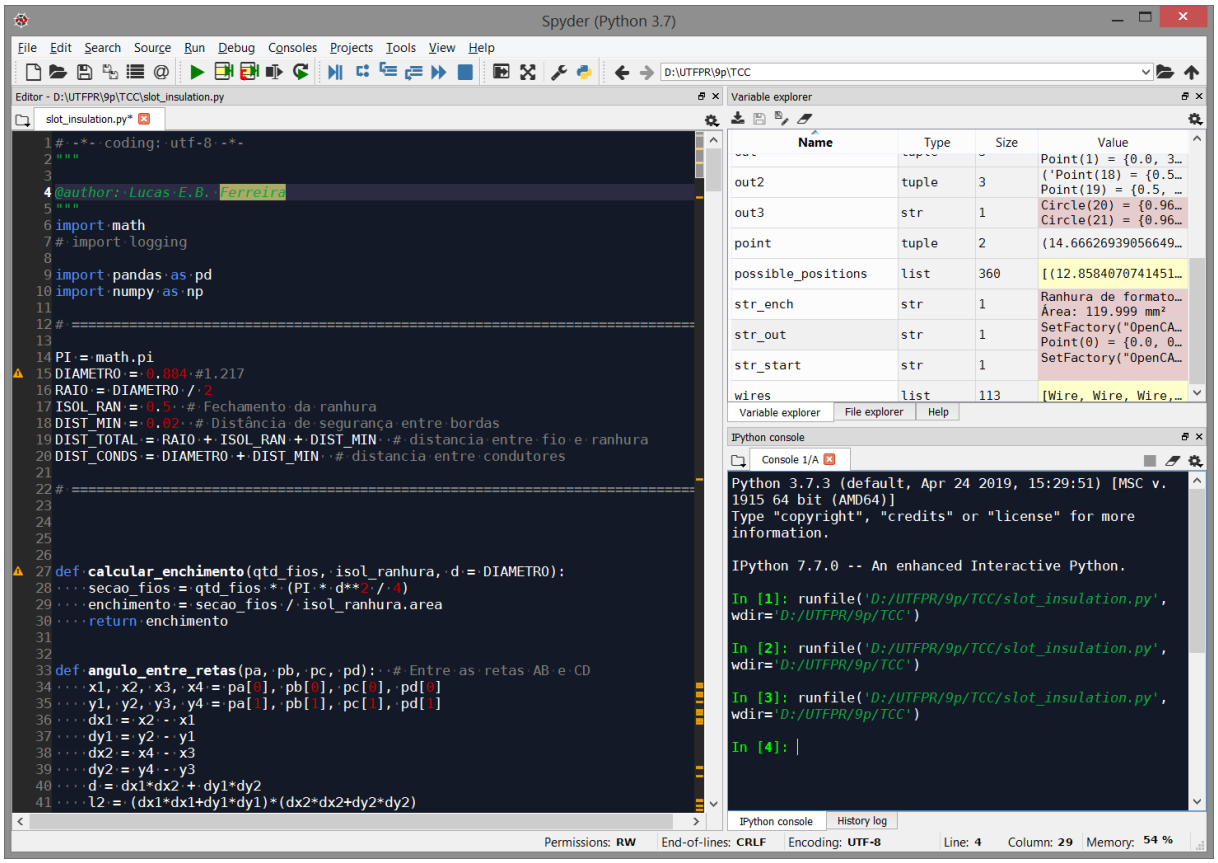


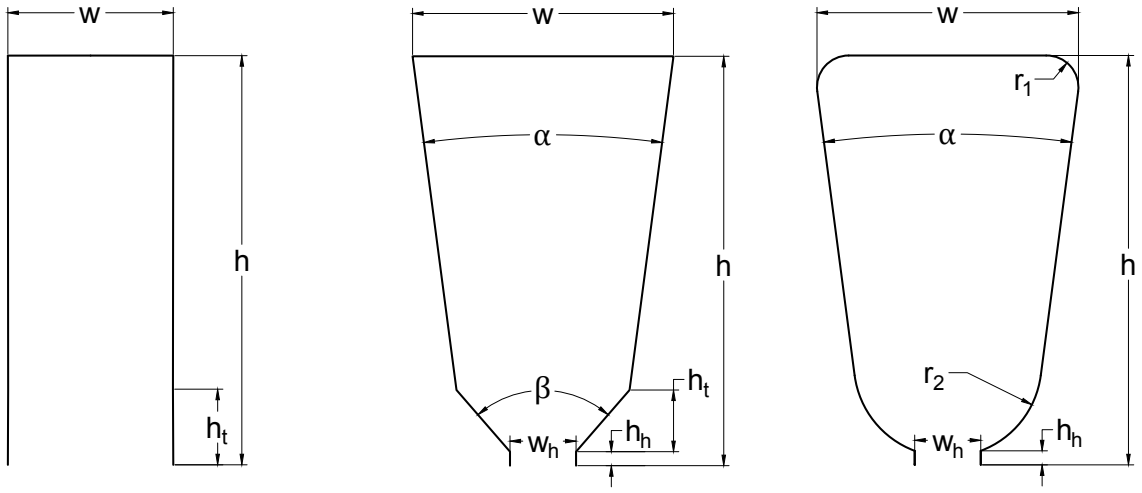
Figura 3 – *Spyder*, um Ambiente de Desenvolvimento Integral
Fonte: Autoria Própria

3.2 MODELAGEM DAS RANHURAS E PARÂMETROS INICIAIS

O algoritmo a ser utilizado, necessita das dimensões geométricas da ranhura do estator para definir a posição dos condutores em seu interior. Faz-se necessário então, definir previamente alguns parâmetros para que possam ser efetuados os devidos cálculos.

Primeiramente, é selecionada a forma geométrica e as dimensões das ranhuras. No trabalho em questão, foram utilizados três modelos, com seus perfis e dimensões representados na Figura 4.

A ranhura representada na Figura 4a apresenta uma modelagem não muito utilizada em projetos de máquinas elétricas, porém, sua determinação nos permite uma aproximação mais simples e didática, de forma a facilitar a implementação inicial do algoritmo, visto que somente as dimensões da profundidade da ranhura, h, a altura do pescoço da ranhura, h_t, e largura, w, são necessárias.



(a) Formato retangular **(b) Formato trapezoidal** **(c) Formato padrão *IEC***

Figura 4 – Formatos de ranhuras utilizados
Fonte: Autoria Própria

Na Figura 4b é apresentado um tipo de ranhura mais comumente utilizado em projetos, baseada na literatura, de modelagem mais complexa que a anterior, porém, por apresentar uma tipologia semi-fechada, permite obter uma redução no *cogging torque*[5] da máquina (HENDERSHOT; MILLER, 2010). Os parâmetros a serem determinados para esta ranhura, conforme demonstrado na figura, são:

- w, a largura total da ranhura;
- w_h, a largura da abertura da ranhura;
- α, o ângulo de abertura da ranhura;
- β, o ângulo de abertura do colarinho da ranhura;
- h, a altura total, ou profundidade da ranhura;
- h_t, a altura do colarinho da ranhura;
- h_h, a altura do pescoço da ranhura;

Por fim, a ranhura da Figura 4c é um modelo padrão de dimensionamento, disponibilizado pela *International Eletrotechnical Comission* para motores de indução trifásicos na carcaça 100 (distância, em mm, do centro do eixo à base de apoio), 2 polos e condutor nominal de 0.80 mm, codificado como *IEC* 100/2.80 (RAABE, 2014; TOMMASO et al., 2017). Este é o mesmo perfil utilizado como base para o algoritmo a

[5] Torque resistente gerado pela interação entre os imãs permanentes do rotor com as ranhuras do estator.

ser implementado. Analogamente aos modelos anteriores, os parâmetros da ranhura a serem predeterminados, apresentados na figura, são:

- w, a largura total da ranhura;
- r_1, o raio dos arcos que compõem os vértices superiores;
- r_2, o raio dos arcos que compõem os vértices inferiores;
- α, o ângulo de abertura da ranhura;
- w_h, a largura da abertura da ranhura;
- h, a altura total, ou profundidade da ranhura;
- h_h, a altura do colarinho da ranhura;

Devido a simetria axial dos três modelos selecionados, a fim de simplificar a modelagem inicial, foi realizado o corte central das ranhuras, rotacionando-as em 90° e representando-as no plano cartesiano em coordenadas x e y, conforme demonstra a Figura 5.

Cada ranhura pode ser representada por um conjunto de pontos, ligados entre si por uma reta, que pode ser descrita pela equação (3) ou arco, descrito por (4).

$$y = a \cdot x + b \tag{3}$$

$$r_{arco}^2 = (x_{arco} - x_0)^2 + (y_{arco} - y_0)^2 \tag{4}$$

Na sequência será descrito o processo realizado para se obter os pontos de interesse das ranhuras e que parâmetros foram considerados.

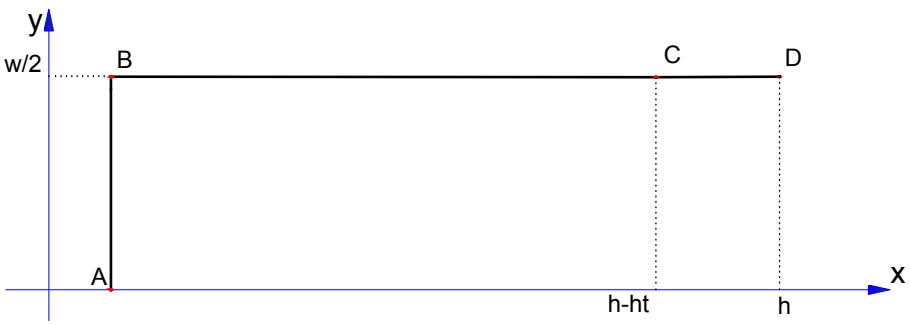


Figura 5 – Ranhura retangular
Fonte: Autoria Própria

A Figura 5 comprova a simplicidade relativa da ranhura retangular, que possui somente 4 pontos, ligados por três retas, $\overline{AB}$, uma reta vertical de coeficiente angular $a_{AB} = \infty$, $\overline{BC}$ e $\overline{CD}$, duas retas horizontais de coeficiente angular $a = 0$

As coordenadas do ponto A são definidas como sendo a origem do plano $x - y$ sendo então definidas como $(0,\ 0)$. O ponto B é obtido através do parâmetro que dita a largura da ranhura, sendo então definido como $(0,\ w/2)$, pelo fato da reta $\overline{AB}$ ser paralela ao eixo y, a coordenada x é igual a zero. O ponto C é consequência da profundidade da ranhura, subtraindo a altura do pescoço da mesma, portanto $(h - h_h,\ w/2)$. Por fim, o ponto D é obtido análogo ao ponto C, através do parâmetro que define a profundidade, temos então $(h,\ w/2)$.

Em resumo, os pontos da ranhura retangular são:

$$\begin{aligned} A: &\quad x_A = 0; && y_A = 0 \\ B: &\quad x_B = 0; && y_B = \frac{w}{2} \\ C: &\quad x_C = h - h_h; && y_C = \frac{w}{2} \\ D: &\quad x_D = h; && y_D = \frac{w}{2} \end{aligned}$$

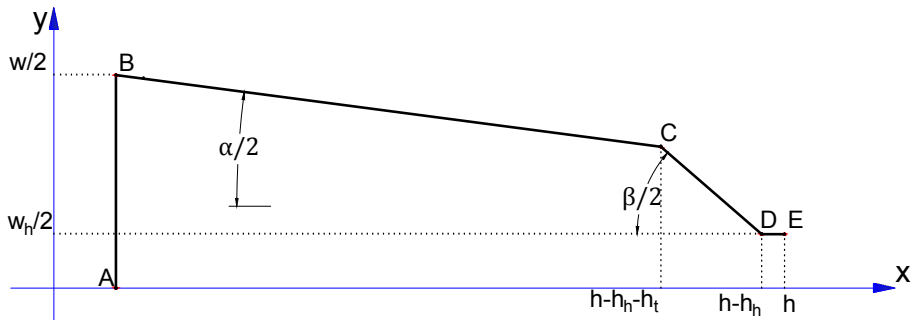


Figura 6 – Ranhura trapezoidal
Fonte: Autoria Própria

A ranhura trapezoidal, mostrada na Figura 6, pode ser representada por 5 pontos, $A - E$, com uma complexidade relativamente maior, tem-se os seguintes parâmetros:

- $\overline{AB}$, uma reta vertical de coeficiente angular $a_{AB} = \infty$;
- $\overline{BC}$, uma reta com coeficiente angular de $a_{BC} = -tan(\frac{\alpha}{2})$;
- $\overline{CD}$, uma reta com coeficiente angular de $a_{CD} = -tan(\frac{\beta}{2})$;
- $\overline{DE}$, uma reta horizontal de coeficiente angular $a_{DE} = 0$.

Para se obter as coordenadas dos pontos desta ranhura, é necessário aliar alguns parâmetros das dimensões da ranhura com as equações das retas $\overline{BC}$ e $\overline{CD}$. Os ponto A, B, D e E são definidos analogamente aos pontos de mesma denominação na ranhura retangular. O cálculo das coordenadas do ponto C pode ser obtido através da análise de uma das retas inclinadas $\overline{BC}$ ou $\overline{CD}$.

Para a coordenada x do ponto C utiliza-se a subtração dos parâmetros que definem a profundidade da ranhura, h, o comprimento do pescoço, h_h e o comprimento do colarinho, h_t. Para a definição do ponto y optou-se por utilizar $\overline{BC}$, sabendo que a mesma cruza o ponto $x = 0$ em $y = y_B$ (coordenada y do ponto B), e que a inclinação é advinda de $\alpha/2$ a partir de (3) é possível definir a equação da reta de (5).

$$y_C = \tan(-\frac{\alpha}{2}) * x_C + y_B \tag{5}$$

Portanto, os pontos da ranhura trapezoidal ficam definidos como:

$$\begin{aligned}
A: &\quad x_A = 0; && y_A = 0 \\
B: &\quad x_B = 0; && y_B = \frac{w}{2} \\
C: &\quad x_C = h - h_h - h_t; && y_C = \tan(-\frac{\alpha}{2}) * x_C + y_B \\
D: &\quad x_D = h - h_h; && y_D = \frac{w_h}{2} \\
E: &\quad x_E = h; && y_E = \frac{w_h}{2}
\end{aligned}$$

A ranhura com os vértices arredondados, vista na Figura 7 é representada por 8 pontos, $A - F$, C_{BC} e C_{DE}, os pontos centrais dos arcos, de raios r_1 e r_2. Esta ranhura possui uma parametrização mais complexa, porém, foi utilizada a representação descrita por Raabe (2014) em conjunto com a análise de Tommaso et al. (2017).

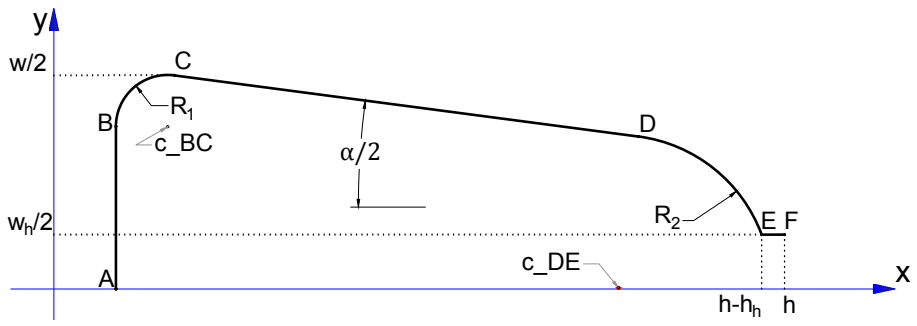


Figura 7 – Ranhura padrão
Fonte: Autoria Própria

A representação de ambos descrevem os seguintes parâmetros:

- $\overline{AB}$, uma reta vertical de coeficiente angular $a_{AB} = \infty$;
- $\widehat{BC}$, um arco de raio r_1 e centro c_{BC};
- $\overline{CD}$, uma reta com coeficiente angular de $a_{CD} = -tan(\frac{\alpha}{2})$;
- $\widehat{DE}$, um arco de raio r_2 e centro c_{DE};
- $\overline{EF}$, uma reta horizontal de coeficiente angular $a_{EF} = 0$.

Além dos parâmetros construtivos descritos acima, foram definidos os pontos principais no plano cartesiano baseando-se nas bibliografias referenciadas.

O ponto A é definido como sendo na origem, padrão tomado como base nas três ranhuras. Para encontrar as coordenadas dos pontos B, C e do centro do arco entre os dois pontos, é necessário analisar a distância r_1 e a correlação com a largura total da ranhura.

Para isso, como pode ser visualizado na Figura 8, foi definido um triângulo retângulo entre o ponto C, o centro c_{BC} do arco 1 e um ponto auxiliar Aux, localizado a uma distância Δy de c_{BC}. O ângulo oposto ao cateto Δx é o mesmo referente à abertura angular da ranhura. O valor de Δy e Δx são encontrados conforme (6) e (7)..

$$\Delta y = r_1 * \cos\frac{\alpha}{2} \tag{6}$$

$$\Delta x = r_1 * \sin\frac{\alpha}{2} \tag{7}$$

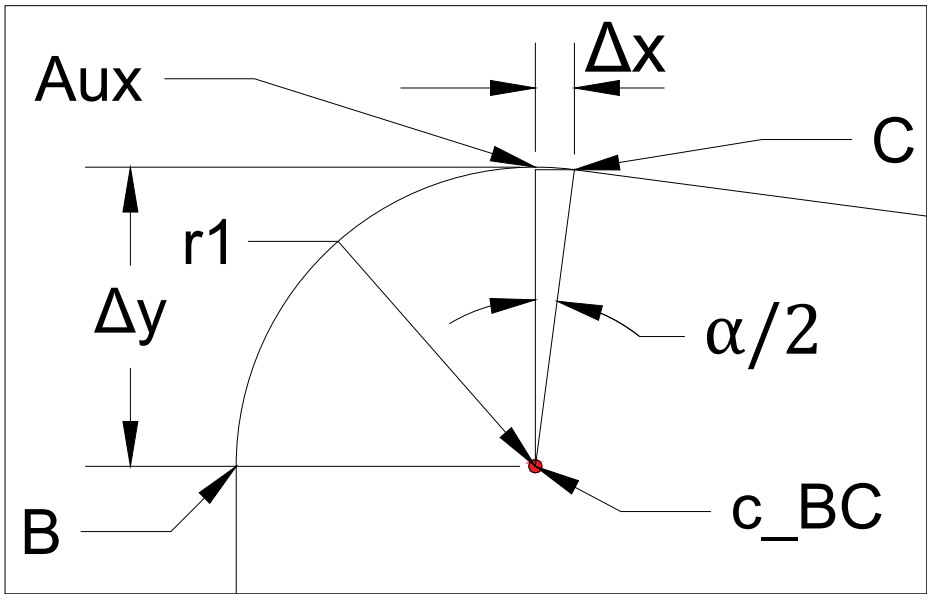


Figura 8 – Visualização do método utilizado para definir os pontos B, C e c_{BC}
Fonte: Autoria Própria

Pelo fato da reta $\overline{AB}$ ser paralela ao eixo y, a coordenada x_B mantém-se 0, e para definirmos a coordenada y_B, subtrai-se Δy de $w/2$, que representa metade da largura total da ranhura. O ponto central do arco $\widehat{BC}$, possui a mesma coordenada em y, com um deslocamento igual ao raio r_1 em x. Para a obtenção do ponto x_C, utiliza-se a variação Δx em relação ao ponto c_{BC}, e o ponto y_C é definido pela largura $w/2$ da ranhura.

O ponto D requer a posição do ponto E definida, que é obtida através da metade da largura total da abertura da ranhura $w_h/2$ e da subtração entre a profundidade e a altura do pescoço da ranhura $h - h_h$, consequentemente, o ponto F é definido em seguida, por ser apenas deslocado h_h no eixo x em relação ao ponto E.

Além disso, no trabalho realizado por Raabe (2014) e Tommaso et al. (2017) é descrita a forma como se obtiveram as coordenadas para o ponto D. Para tal modelagem, foi necessário definir a equação da reta $\overline{CD}$, de forma semelhante a (5), obtém-se os parâmetros de $\overline{CD}$ demonstrados em (8) e (9).

$$a_{CD} = \tan\Big(-\frac{\alpha}{2}\Big) \tag{8}$$

$$b_{CD} = y_C - a_{CD} * x_C \tag{9}$$

A partir das equações da reta (3) e do arco (4), um equacionamento aplicado aos parâmetros definidos acima leva a definição do ponto x_D. A equação resulta em duas possibilidades, com a definida por (10) sendo a única aplicável ao caso estudado (RAABE, 2014).

$$x_D = -\frac{p}{2} - \sqrt{\Big(\frac{p}{2}\Big)^2 - q} \tag{10}$$

Com p e q definidos em (11) e (12).

$$p = -2 * \frac{b_{CD} * y_1 + x_1}{b_{CD}^2 + 1} \tag{11}$$

$$q = \frac{y_1^2 + x_1^2 - r_2^2}{b_{CD}^2 + 1} \tag{12}$$

E as duas variáveis de suporte, x_1 e y_1 definidas em (13) e (14).

$$x_1 = r_2 * \sin\Big(\frac{\alpha}{2}\Big) + x_E \tag{13}$$

$$y_1 = y_E - b_{CD} + r_2 * \cos\Big(\frac{\alpha}{2}\Big) \tag{14}$$

Já o ponto y_D é advindo da equação (9), conforme pode ser visualizado em (15).

$$y_D = a_{CD} * x_D + b_{CD} \tag{15}$$

Por fim, são calculadas as coordenadas do ponto central do arco $\widehat{DE}$, conforme descritas no trabalho de Tommaso et al. (2017) e demonstradas em (16) e (17).

$$x_{c,DE} = x_D - r_2 * \sin\left(\frac{\alpha}{2}\right) \tag{16}$$

$$y_{c,DE} = y_D - r_2 * \cos\left(\frac{\alpha}{2}\right) \tag{17}$$

Em resumo, os pontos referentes à ranhura padrão são definidos como sendo:

$$\begin{aligned}
A: &\quad x_A = 0; && y_A = 0\\
B: &\quad x_B = 0; && y_B = \frac{w}{2} - r_1 * \cos\left(\frac{\alpha}{2}\right)\\
c_{BC}: &\quad x_{c,BC} = r_1; && y_{c,BC} = y_B\\
C: &\quad x_C = x_{c,BC} + r_1 * \sin\left(\frac{\alpha}{2}\right); && y_C = \tan\left(-\frac{\alpha}{2}\right) * x_C + y_B\\
D: &\quad x_D = -\frac{p}{2} - \sqrt{\left(\frac{p}{2}\right)^2 - q}; && y_D = \tan\left(-\frac{\alpha}{2}\right) * x_D + y_B\\
c_{DE}: &\quad x_{c,DE} = x_D - r_2 * \sin\left(\frac{\alpha}{2}\right); && y_{c,DE} = y_D - r_2 * \cos\left(\frac{\alpha}{2}\right)\\
E: &\quad x_E = h - h_h; && y_E = \frac{w_h}{2}\\
F: &\quad x_F = h; && y_F = \frac{w_h}{2}
\end{aligned}$$

3.3 CÁLCULO DAS ÁREAS DAS RANHURAS

Para o cálculo do fator de preenchimento de ranhura, é necessário ter conhecimento, além da seção dos condutores, da área da ranhura que se está projetando. Para encontrar este parâmetro, foram utilizados alguns métodos descritos na sequência.

Para a ranhura retangular, o cálculo de sua área é trivial, somente sendo necessário calcular o produto entre a largura w e a altura h.

Já na ranhura trapezoidal, foi feita a divisão da parte interna em três triângulos, $A1$, $A2$ e $A3$, para que suas áreas fossem somadas afim de obter a área total, esse seccionamento pode ser visualizado na Figura 9.

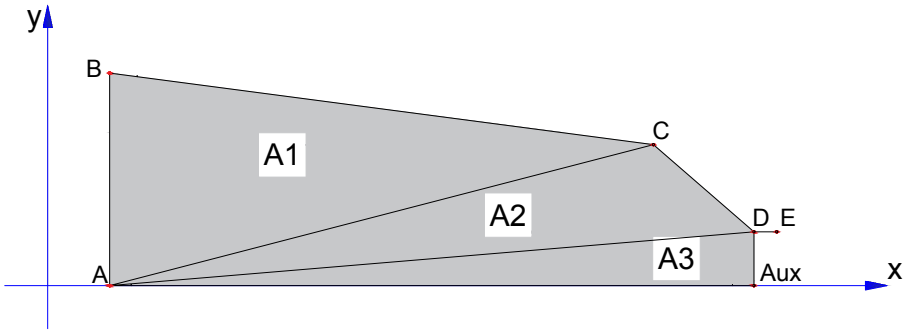


Figura 9 – Divisão interna das áreas da ranhura trapezoidal
Fonte: Autoria Própria

As áreas dos triângulos foram calculadas utilizando a equação (18), aonde a área de um triângulo $\widehat{ABC}$ é calculada em função das coordenadas dos seus vértices.

$$Area_{\widehat{ABC}} = |x_A * (y_B - y_C) + x_B * (y_C - y_A) + x_C * (y_A - y_B)| \tag{18}$$

Para a ranhura padrão, o método de cálculo segue o mesmo princípio da trapezoidal, subdividindo as áreas internas em geometrias mais simples para no final obter o total a partir da soma. Neste caso, a divisão foi feita em 8 seções, desconsiderando a parte pertencente à abertura da ranhura (abaixo da reta $\overline{EF}$), conforme demonstrado na Figura 10.

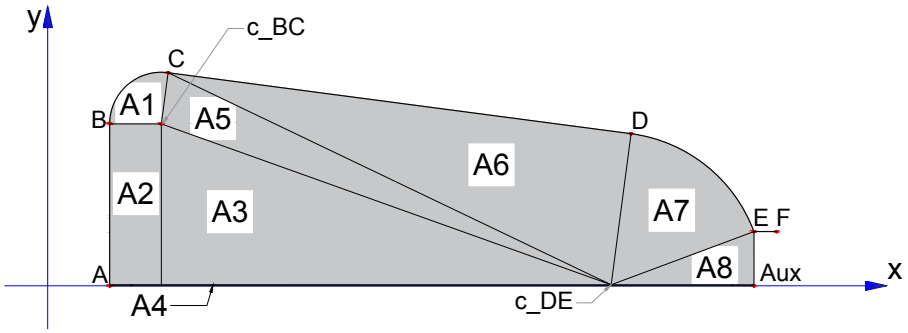


Figura 10 – Divisão interna das áreas da ranhura padrão
Fonte: Autoria Própria

Aonde:

- A1: Arco $\widehat{BC}$ de raio r_1;
- A2: Retângulo de base r_1 e altura $y_B - y_{c,DE}$;
- A3: Triângulo entre os pontos c_{BC}, c_{DE} e $[x_{c,BC}; y_{c,DE}]$;
- A4: Retângulo de base x_E e altura $y_{c,DE}$;

- A5: Triângulo entre os pontos $c_{BC},\ C,$ e c_{DE};
- A6: Triângulo entre os pontos $C,\ D,$ e c_{DE};
- A7: Arco $\widehat{DE}$ de raio r_2;
- A8: Triângulo entre os pontos E, c_{DE} e $[x_E; y_{c,DE}]$.

A partir do ângulo entre as retas formadas entre os pontos B e C até o centro c_{BC}, para r_1 conforme demonstrado em (19) (WINTERLE, 2000), é possível obter a área do arco 1 na equação 20, e de forma análoga para r_2 .

$$den = (x_{c,BC} - x_B) * (x_{c,BC} - x_C)) + ((y_{c,BC} - y_B) * (y_{c,BC} - y_C)$$
$$num = ((x_{c,BC} - x_B)^2 + (y_{c,BC} - y_B)^2) * ((x_{c,BC} - x_C)^2 + (y_{c,BC} - y_C)^2)$$

$$\theta_{arco} = \cos^{-1}\left(\frac{den}{num}\right) \tag{19}$$

$$A_{arco} = \frac{1}{2} * r_1^2 * \theta_{arco,rad} \tag{20}$$

Posteriormente, para obter a área total útil da ranhura, o valor das subseções é multiplicado por 2 para abranger a parte inferior da ranhura, e é então subtraída a área referente ao material isolante inserido entre a ranhura e os condutores.

3.4 DEFINIÇÃO DOS CONTORNOS E DISTÂNCIAS

Para realizar as manipulações dos dados referentes aos contornos das ranhuras e posicionamento dos condutores em seu interior, é preciso implementar condições que determinam a distância entre os objetos analisados.

Os condutores são definidos por seu ponto central, seu diâmetro e um valor adicional para representar o material isolante ao seu redor. A IEC (2013) fornece os dados referentes aos diâmetros padrões e qual o acréscimo a ser considerado pelo isolante. Diferentes diâmetros de condutores serão utilizados para que os objetivos do trabalho sejam alcançados.

A definição completa da ranhura é determinada pelos pontos principais, obtidos em 3.2, e pelo espelhamento em relação ao eixo x dos mesmos. As condições de contorno são determinadas pelas retas e/ou arcos que os ligam.

Para implementar a interação entre os condutores e as ranhuras, a distância entre as bordas e os condutores deve ser definida para os três perfis estudados, de

forma que ao serem posicionados os centros dos condutores, pelo algoritmo, não haja sobreposição de objetos, nem a inserção em pontos externos à ranhura.

Dois fatores adicionais que foram considerados para os três casos são:

- D_{min}: mínima distância entre o condutor e suas limitações no entorno, podendo esta ser as paredes da ranhura ou outro condutor. Esse parâmetro assegura que não haverá uma distância nula entre objetos, algo impraticável fisicamente;
- D_{isol}: representa o isolante de fundo de ranhura, ou de fechamento, que é inserido para evitar o contato elétrico entre os condutores e as chapas do estator.

Para determinar uma posição válida para os condutores, são necessários os conceitos de distância entre pontos e entre um ponto e uma reta.

A equação (21) define a distância entre um ponto P, de coordenadas (P_x, P_y) e um ponto Q, de coordenadas (Q_x, Q_y). Já na equação (22), é calculada a menor distância do ponto P a uma reta definida por dois pontos, A e B (WINTERLE, 2000).

$$D(P,Q) = \sqrt{(Q_x - P_x)^2 + (Q_y - P_y)^2} \tag{21}$$

$$D(P,\overline{AB}) = \frac{P_x * (B_y - A_y) - P_y * (B_x - A_x) + B_x * A_y - B_y * A_x}{\sqrt{(B_y - A_y)^2 + (B_x - A_x)^2}} \tag{22}$$

O que garantirá que não sejam posicionados condutores sobrepostos, é a definição da distância mínima entre o ponto central dos dois, que é caracterizada pela soma entre o diâmetro do fio, com isolante, e o fator D_{min}.

As definições necessárias para verificar se um condutor está ou não dentro da ranhura são compostas por uma série de checagens que variam, em sua parametrização, de acordo com o perfil desta. Na sequência são descritas essas verificações para cada tipo de ranhura.

3.4.1 Condições de Contorno da Ranhura Retangular

No caso de uma ranhura de contorno retangular, a condição para o posicionamento interno é limitada pelas coordenadas $x : (x_A \rightarrow x_C)$ e $y : (y_E \rightarrow y_B)$. Simplificadamente, não se faz necessário o uso da equação (22) devido ao fato das limitantes serem paralelas aos eixos. As distâncias do entorno podem ser visualizadas na Figura 11.

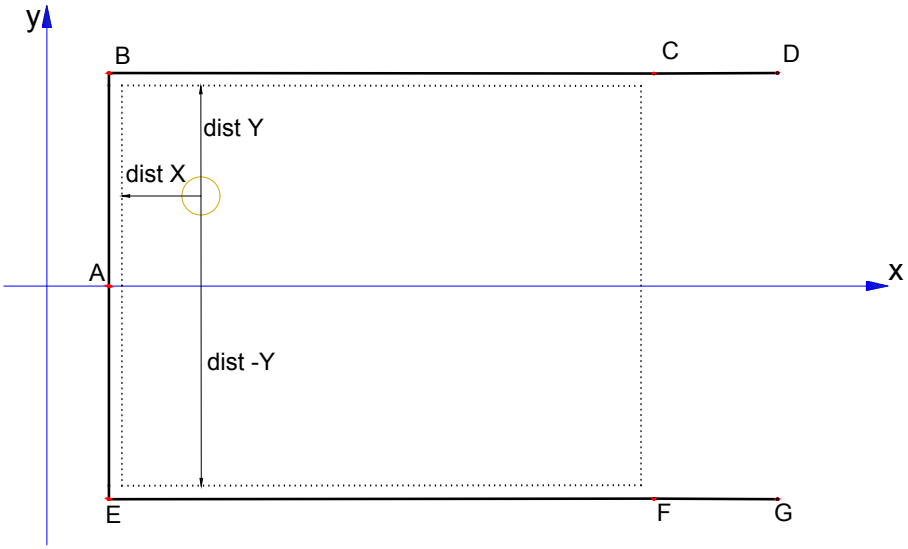


Figura 11 – Condição de contorno na ranhura retangular
Fonte: Autoria Própria

As três posições, $distX$, Y e $-Y$ são calculadas a partir da subtração entre as coordenadas limitantes e o centro do condutor. A condição para a posição ser validada é que esses comprimentos devem ser inferiores a soma entre o raio do condutor, D_{min} e D_{isol}.

3.4.2 Condições de Contorno da Ranhura Trapezoidal

Para a ranhura de contorno trapezoidal, faz-se necessário calcular a proximidade entre o condutor e as diversas retas que a compõem.

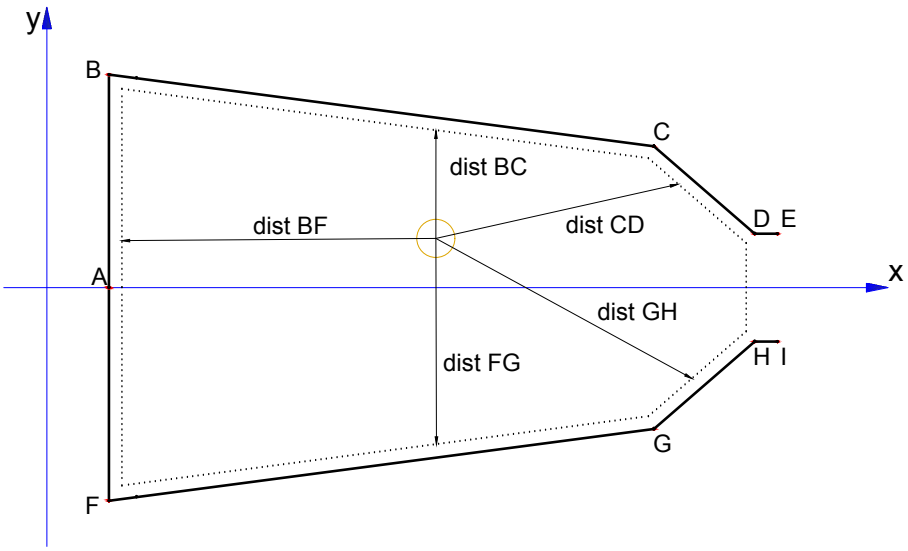


Figura 12 – Condição de contorno na ranhura trapezoidal
Fonte: Autoria Própria

Analogamente à retangular, as dimensões do isolante e da distância mínima são consideradas para compor a condição de posicionamento. Na Figura 12 são demonstradas quais distâncias devem ser calculadas para utilização do algoritmo.

3.4.3 Condições de Contorno da Ranhura Padrão

No caso da ranhura padrão, o método para impor as condições limitantes tem como base a subdivisão interna da ranhura em setores conforme pode ser visto na Figura 13.

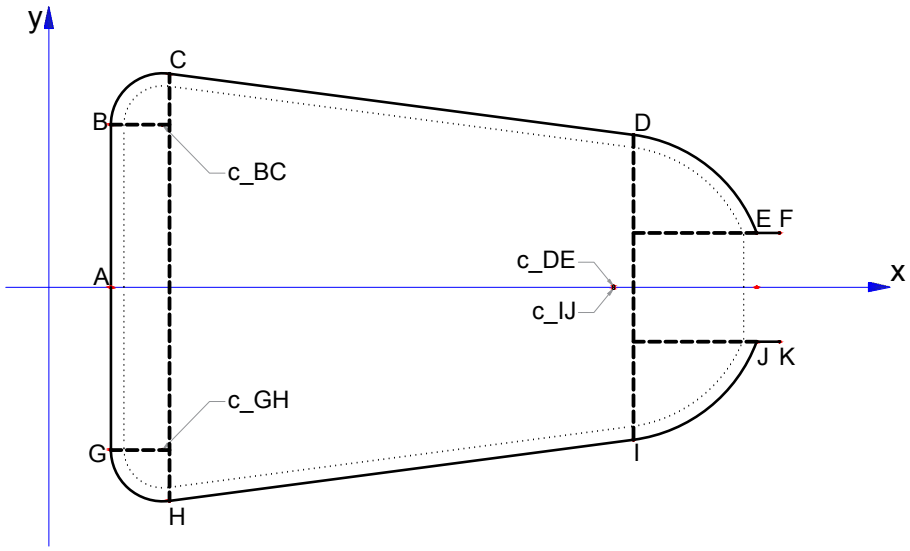


Figura 13 – Setores internos da ranhura padrão
Fonte: Autoria Própria

Cada setor possui um *set* de condições para definir suas bordas e distâncias a serem respeitadas e são divididas em domínios que dependem da coordenada x, ou y limitante, a geometria e a tratativa para cada uma são:

- Retangular: Dois setores retangulares, um localizado entre os pontos G, B, c_{GH} e c_{BC} e o outro limitado por $x : (x_D \rightarrow x_E)$ e $y : (y_J \rightarrow y_E)$, tornando sua tratativa semelhante à ranhura retangular;
- Trapezoidal: O contorno trapezoidal é delimitado pelos pontos C, D, H e I e suas condições são análogas as da ranhura trapezoidal;
- Arcos: As regiões que englobam os contornos arredondados da ranhura são limitadas pelos inverso das regiões anteriores, para definir seu contorno é calculada a distância entre o ponto central do arco e do condutor, conforme pode ser visualizado na Figura 14, o valor de r_{dist} deve ser menor ou igual a subtração entre o comprimento do raio e os fatores padrões (isolante e distância mínima).

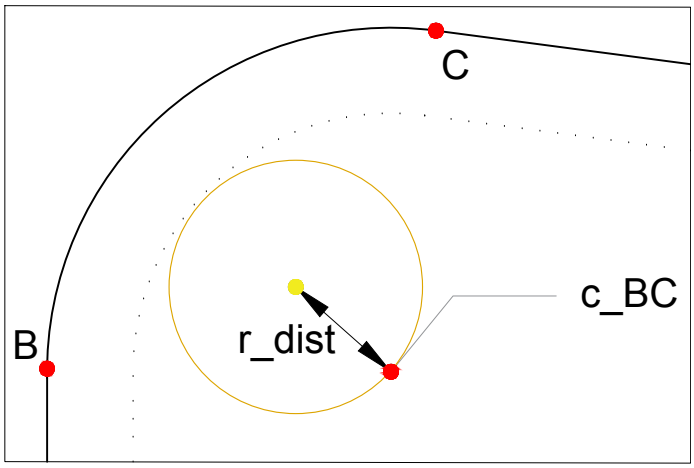


Figura 14 – Ranhura com isolante e posição inicial de condutores
Fonte: Autoria Própria

3.5 IMPLEMENTAÇÃO DO ALGORITMO

Com a parametrização dos contornos das ranhuras e as condições necessárias para que não haja sobreposição de condutores, o algoritmo realizará uma sequência de etapas que irão, iterativamente, realizar o posicionamento do máximo possível de condutores no interior da ranhura.

Esse valor máximo possui variação dependendo da posição do primeiro condutor. Tendo isso em vista, foram selecionadas três posições iniciais, com o objetivo de analisar essa variação. Essas posições são são mostradas na Figura 15 e descritas como:

1. Ponto mais a esquerda e ao centro da ranhura;
2. Ponto no canto superior a esquerda;
3. Ponto no canto inferior a esquerda.

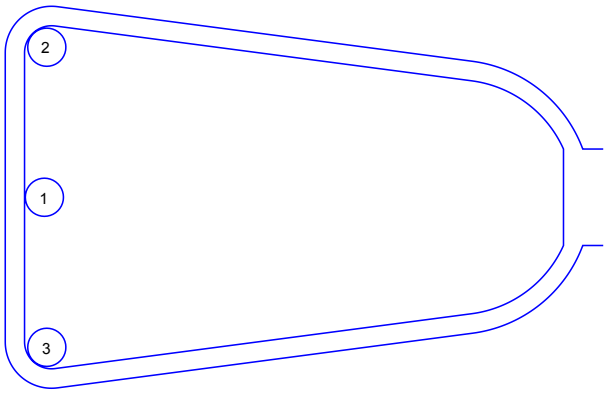


Figura 15 – Posição inicial dos condutores na ranhura
Fonte: Autoria Própria

O algoritmo completo pode ser visualizado no Apêndice A. De forma resumida, a sequência de operações ocorre da seguinte forma:

1. Determinam-se a dimensão da folha isolante de fundo de ranhura, e de fechamento, anexa às extremidades da ranhura;
2. Insere-se o primeiro condutor, de diâmetro d_c na posição inicial 1;
3. É verificado um perímetro, com seu raio determinado pela soma entre d_c e o fator D_{min}, gerado a partir da equação (23), com t variando de $0 \rightarrow 2\pi$, e o ponto $P = (x_c, y_c)$ determinado para as diferentes posições (1, 2 ou 3) do centro deste condutor;

$$P_{circulo} = (R * \cos t + x_c)\ \vec{i} + (R * \sin t + y_c)\ \vec{j} \tag{23}$$

4. Guardam-se os pontos de (23) em uma lista. Os pontos são ordenados de forma que a sequência para percorrer esta lista seja de π até 0, no sentido anti-horário, e posteriormente no sentido horário;
5. Percorre-se a lista, sequencialmente, e são verificados os pontos que permitem a inserção de um condutor, determinado por duas condições:
   a) Estar dentro da ranhura; e
   b) Não sobrepor outro condutor.
6. Após percorrer a lista de pontos para o primeiro condutor, é realizada a finalização deste ponto, e então passa-se para o próximo ponto válido gerado;
7. Repetem-se estes passos até que todos os pontos estejam marcados como finalizados;
8. Reinicia-se o processo com uma nova posição, 2 ou 3, para o primeiro condutor;
9. Por fim, após todas as soluções possíveis serem determinadas, é feita uma comparação entre elas e escolhe-se a de maior fator de preenchimento de ranhura, ou menor para uma aproximação mais conservadora (RAABE, 2014).

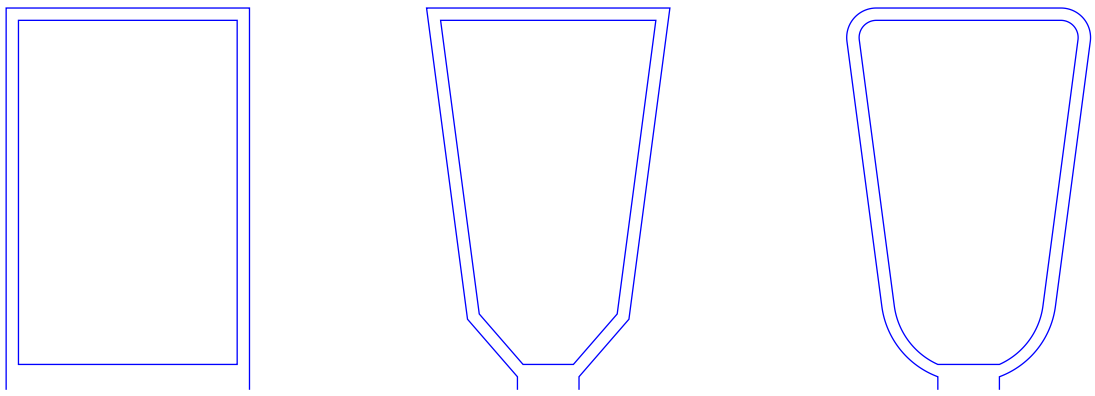
4 RESULTADOS E DISCUSSÃO

Neste capítulo são apresentados os dados e as análises referentes aos resultados da implementação do algoritmo; as condições que foram consideradas para cada um dos casos a serem analisados; bem como as conclusões que podem ser retiradas do trabalho desenvolvido. Também são apresentadas sugestões que podem ser utilizadas em trabalhos futuros que sigam a mesma linha de atuação.

4.1 CARACTERÍSTICAS E DIMENSÕES DAS RANHURAS E DOS CONDUTORES

Para que fosse possível realizar a verificação da efetividade e aplicabilidade do algoritmo, foram realizadas diversas implementações com características e dimensões variadas. Em relação as ranhuras, além dos três formatos já descritos em 3.2, foram utilizadas duas parametrizações para cada formato: uma de perfil mais alongado, com uma área maior, e outra mais curta, com a largura mais aproximada do comprimento total.

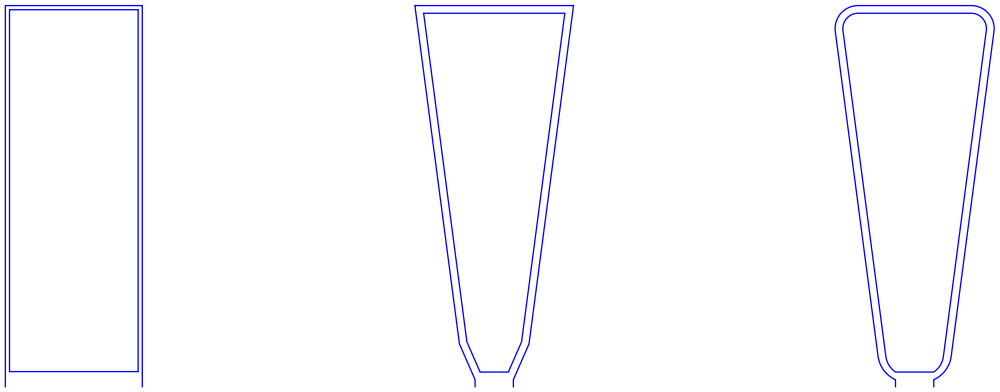
Os parâmetros da ranhura padrão de perfil curto, observada na Figura 16c, são baseados nos modelos da norma *IEC* e foram adaptados a partir do trabalho de Raabe (2014), para poder se ter uma base nos resultados obtidos pelo autor do algoritmo original. Além disso, os valores foram utilizados também para as ranhuras retangular e trapezoidal, mostradas nas Figuras 16a e 16b, respectivamente, com exceção dos valores dos raios dos arcos, que não são aplicáveis a esses formatos.



(a) Formato retangular **(b) Formato trapezoidal** **(c) Formato padrão**

Figura 16 – Ranhuras de perfil curto
Fonte: Autoria Própria

O perfil alongado foi obtido através da adaptação do trabalho de Caruso et al. (2018), e também teve seus parâmetros exportados para as ranhuras de formatos não-padronizado, visualizável nas Figuras 17a, 17b e 17c.



(a) Formato retangular **(b) Formato trapezoidal** **(c) Formato padrão**

Figura 17 – Ranhuras de perfil alongado
Fonte: Autoria Própria

Os valores numéricos utilizados no dimensionamento das ranhuras são descritos na Tabela 1.

Tabela 1 – Parâmetros dimensionais das ranhuras

Parâmetro	Unidade	Variável	Perfil padrão	Perfil alongado
Largura	mm	w	9,88	10,36
Profundidade	mm	h	15,50	24,93
Abertura	mm	w_h	2,50	2,50
Altura da abertura	mm	h_h	0,53	0,50
Altura do colarinho	mm	h_t	2,3351	2,3351
Ângulo de abertura	$Graus$	α	15	15
Raio do arco superior	mm	r_1	1,20	1,4998
Raio do arco inferior	mm	r_2	3,54	1,9711

A área de cada ranhura, necessária para a obtenção do fator de enchimento, foi calculada através da tratativa descrita na Seção 3.3. Esses valores são apresentados na Tabela 2.

A área útil é obtida desconsiderando a parte referente a abertura da ranhura, que não permite a inserção de condutores, e a seção preenchida pela folha isolante.

Tabela 2 – Áreas das ranhuras

Ranhura		Área mm^2	Área útil mm^2
Curta	Retangular	147.90	124.05
	Trapezoidal	114.39	93.31
	Padrão	120.00	99.07
Longa	Retangular	253.10	219.31
	Trapezoidal	172.86	142.56
	Padrão	183.33	153.13

Foram utilizadas cinco diferentes bitolas para os condutores do cálculos de fator de enchimento, para se obter uma análise com dados diversificados. Os condutores utilizados e suas dimensões são apresentadas na Tabela 3 (IEC, 2013).

Tabela 3 – Parâmetros dimensionais dos condutores

Diâmetro nominal mm	Diâmetro isolado mm
0,71	0,789
0,80	0,884
0,90	0,989
1,00	1,094
1,12	1,217

O valor para o diâmetro isolado do condutor é referente ao maior diâmetro, estabelecido na norma, quando utilizado o segundo grau de isolação (IEC, 2013).

Adicionalmente, para a espessura da folha isolante do fechamento de ranhura, foi considerado um valor de $0,5\ mm$ e para a menor distância possível entre objetos, D_{min} foi pré-determinado um valor de $0,02\ mm$ (RAABE, 2014).

4.2 ANÁLISE DA RANHURA RETANGULAR

A implementação do algoritmo na ranhura retangular foi tida como o ponto de partida para o desenvolvimento da modelagem e da forma com que seriam avaliados os outros tipos de ranhura. Conforme descrito no capítulo 3, essa geometria, visivelmente simples, têm aplicabilidade reduzida quando comparada com as demais.

A desvantagem no seu uso se deve ao fato de causarem um aumento nos harmônicos e no *cogging torque* da máquina (HENDERSHOT; MILLER, 2010). Porém, seu uso torna-se vantajoso em máquinas de maior potência e tensão elevada, aonde se utilizam bobinas pré-formadas, com condutores de formato retangular.

4.2.1 Ranhura Retangular de Perfil Curto

Com esta primeira implementação, foram obtidos os resultados dispostos na Tabela 4.

Tabela 4 – Retangular curta

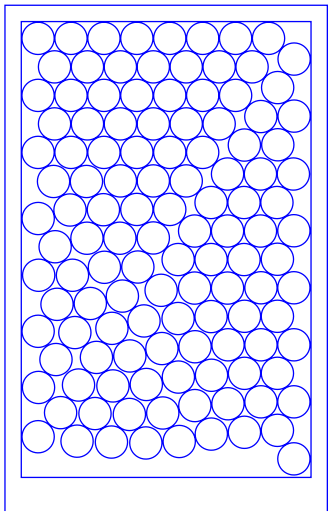
Diâmetro dos condutores mm	Número de condutores	Enchimento cobre %	Enchimento total %	Posição do primeiro condutor
0,789	186	59,36	73,31	1
	185	59,04	72,91	2
	185	59,04	72,91	3
0,884	144	58,35	71,24	1
	148	59,97	73,22	2
	148	59,97	73,22	3
0,989	119	61,03	73,69	1
	116	59,49	71,83	2
	121	62,05	74,93	3
1,094	95	60,15	71,98	1
	96	60,78	72,74	2
	96	60,78	72,74	3
1,217	79	62,74	74,08	1
	75	59,56	70,33	2
	78	61,95	73,14	3

Devido ao fato das dimensões da ranhura retangular terem sido baseadas da ranhura padrão, a sua área acabou sendo relativamente maior. Entretanto, o fator de preenchimento para essa ranhura apresentou valores significativamente bons, demonstrado pelo fato de não haver nenhum cálculo abaixo de 70 %, um valor considerado alto por Hendershot e Miller (2010).

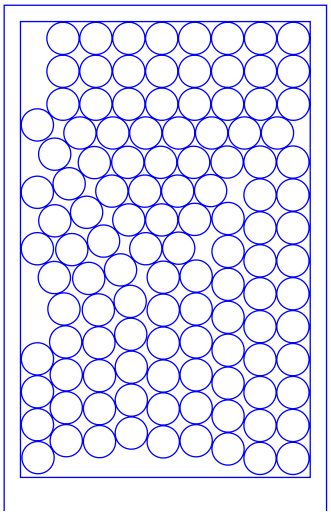
O maior valor calculado para a ranhura retangular curta foi de 74,93%, com um total de 121 condutores de 0,9 mm ($d_{isol} = 0,989\ mm$), a partir da posição inicial 3. Para efeito de comparação, o menor valor encontrado para esse condutor foi de 71,83%, através da posição inicial 2, onde houve uma diminuição de 5 fios, o que reforça a importância da posição inicial no resultado final.

O melhor resultado para essa geometria pode ser visualizado na Figura 18a. A comparação realizada para um diferente posicionamento do primeiro condutor é apresentada na Figura 18b.

Um ponto a se notar é o padrão formado na ranhura com o maior enchimento encontrado, a organização dos condutores assume uma geometria hexagonal. Este posicionamento é causado pelo método utilizado para a inserção dos condutores no



(a) 121 condutores, Fator: 74,93%, Diâmetro: 0,989 mm, Posição: 3



(b) 116 condutores, Fator: 71,83%, Diâmetro: 0,989 mm, Posição: 2

Figura 18 – Resultados do algoritmo para a ranhura retangular curta
Fonte: Autoria Própria

modelo, descrito nos itens 3 e 4 da sequência do algoritmo, visto na Seção 3.5. Esta formação é considerada uma das formas mais otimizadas de se preencher um espaço retangular com círculos de mesmas dimensões (TOMMASO et al., 2017).

4.2.2 Ranhura Retangular de Perfil Longo

Com a segunda implementação da ranhura retangular foram obtidos os resultados dispostos na Tabela 5.

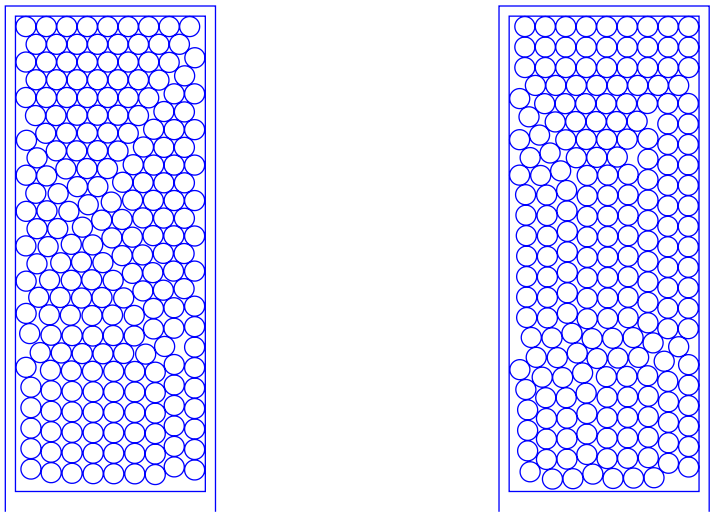
A partir dos dados referentes a ranhura mais alongada, conclui-se que essa característica proporciona um leve aumento no fator de preenchimento. Quando utilizados fios de diâmetro menor, percebe-se um maior aumento relativo. Nesta comparação, na medida que o diâmetro aumenta, esse valor converge para uma média mais próxima entre os valores de enchimento da ranhura curta e da alongada.

De forma semelhante à ranhura curta, o maior enchimento obtido foi para o o condutor de diâmetro 0,9 mm, com o condutor inicial na posição 3. A redução obtida em relação ao menor fator de enchimento para este condutor, nesse caso, foi de 2 fios.

O melhor resultado obtido para esse caso é representado na Figura 19a e a comparação realizada para um diferente posicionamento do primeiro condutor é apresentada na Figura 19b.

Tabela 5 – Retangular longa

Diâmetro dos condutores mm	Número de condutores	Enchimento cobre %	Enchimento total %	Posição do primeiro condutor
0,789	329	59.40	73.35	1
	329	59.40	73.35	2
	333	60.12	74.24	3
0,884	265	60.74	74.16	1
	264	60.51	73.88	2
	267	61.20	74.72	3
0,989	210	60.92	73.56	1
	209	60.63	73.21	2
	216	62.66	75.66	3
1,094	169	60.52	72.44	1
	171	61.24	73.30	2
	171	61.24	73.30	3
1,217	134	60.20	71.08	1
	133	59.75	70.55	2
	142	63.79	75.32	3



(a) 216 condutores, Fator: 75,66%, Diâmetro: 0,989 mm, Posição: 3

(b) 209 condutores, Fator: 73,21%, Diâmetro: 0,989 mm, Posição: 2

Figura 19 – Resultados do algoritmo para a ranhura retangular longa
Fonte: Autoria Própria

4.3 ANÁLISE DA RANHURA TRAPEZOIDAL

A ranhura trapezoidal foi o próximo passo no aperfeiçoamento e na configuração do algoritmo. A geometria desta ranhura garante um melhor desempenho para a máquina

elétrica, devido ao seu aspecto mais fechado, como descrito por (HENDERSHOT; MILLER, 2010).

Essas características adicionais em relação a ranhura retangular formam uma camada de complexidade extra na modelagem de seus parâmetros e na forma como o posicionamento dos condutores interage com os contornos da ranhura, e com outros condutores, durante o processo de inserção.

Analogamente ao formato anterior, foram realizadas duas implementações referente ao perfil da ranhura, curto ou alongado, e para cada um, foram analisadas 3 posições iniciais dos condutores de diâmetros variados.

4.3.1 Ranhura Trapezoidal de Perfil Curto

Os resultados obtidos com a aplicação do algoritmo para as condições estabelecidas pela ranhura trapezoidal de perfil curto foram relacionados na Tabela 6.

Tabela 6 – Trapezoidal curta

Diâmetro dos condutores mm	Número de condutores	Enchimento cobre %	Enchimento total %	Posição do primeiro condutor
0,789	136	57,71	71,26	1
	136	57,71	71,26	2
	136	57,71	71,26	3
0,884	108	58,18	71,04	1
	108	58,18	71,04	2
	108	58,18	71,04	3
0,989	87	59,32	71,63	1
	88	60,00	72,45	2
	87	59,32	71,63	3
1,094	72	60,60	72,53	1
	72	60,60	72,53	2
	71	59,76	71,53	3
1,217	55	58,07	68,57	1
	56	59,13	69,81	2
	57	60,18	71,06	3

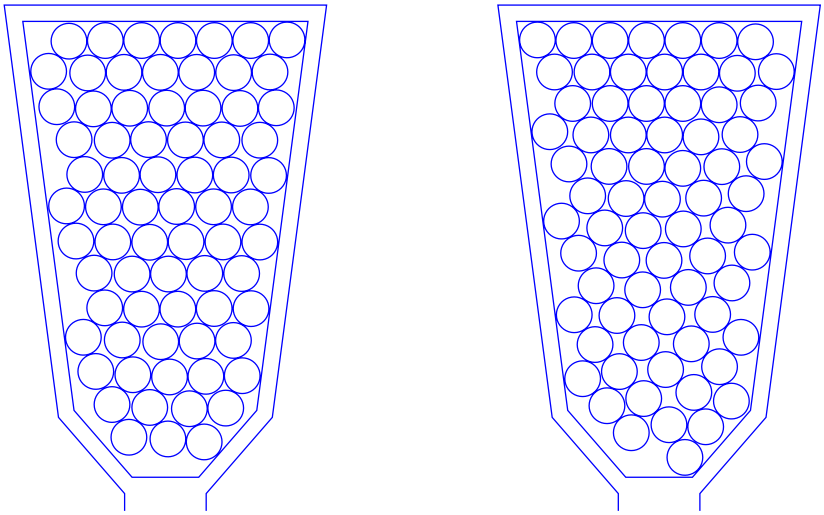
O perfil trapezoidal, como possível observar nos resultados, sofreu uma redução não muito expressiva no fator de enchimento quando comparada com a ranhura de geometria retangular. Essa redução, bem como a redução no número de condutores que pode-se inserir na ranhura, era algo já esperado pois, apesar de possuírem as mesmas dimensões absolutas (altura e largura), a ranhura trapezoidal insere limitações que tornam o processo de bobinagem um pouco mais complexo. Ainda assim, os

valores obtidos são relativamente altos e reforçam a aplicabilidade do algoritmo no quesito de determinar a otimização para a geometria imposta.

Um aspecto particular que ocorreu nesta implementação, foi o fato do maior fator de preenchimento encontrado para os condutores 0,71 mm ($d_{isol} = 0,789\ mm$) e 0,80 mm ($d_{isol} = 0,884\ mm$) não ser alterado pela posição inicial do primeiro condutor, tendo sido mantido em 71,26% e 71,04%, respectivamente para as três inserções do primeiro condutor.

Todavia, para o maior enchimento encontrado, utilizando o condutor de 1,00 mm ($d_{isol} = 1,094\ mm$) foi possível alcançar 72,53% de preenchimento total na ranhura, nas posições 1 e 2. A posição 3, para comparação, apresentou um enchimento de 71,53%, com uma redução de apenas 1 fio.

O melhor caso para esta geometria pode ser observado na Figura 20a e para comparação, o caso com redução de 1 fio devido à mudança na posição do primeiro condutor é visto na Figura 20b.



(a) 72 condutores, Fator: 72,53%, Diâmetro: 1,094 mm, Posição: 2

(b) 71 condutores, Fator: 71,53%, Diâmetro: 1,094 mm, Posição: 3

Figura 20 – Resultados do algoritmo para a ranhura trapezoidal curta
Fonte: Autoria Própria

4.3.2 Ranhura Trapezoidal de Perfil Longo

Para a ranhura com dimensões alongadas, e de formato trapezoidal, foram obtidos os seguintes resultados, descritos na Tabela 7.

O padrão obtido nos dados referentes a esta implementação se mantém semelhante ao perfil curto. Aqui se constatou também que, como o caso anterior, a posição inicial do primeiro condutor alterou de forma pouco significativa o resultado final.

Tabela 7 – Trapezoidal longa

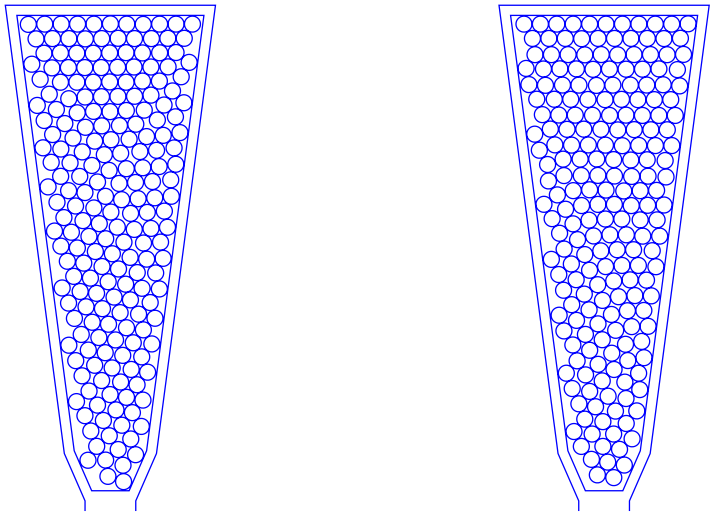
Diâmetro dos condutores mm	Número de condutores	Enchimento cobre %	Enchimento total %	Posição do primeiro condutor
0,789	212	58.88	72.71	1
	210	58.32	72.02	2
	210	58.32	72.02	3
0,884	165	58.18	71.04	1
	168	59.24	72.33	2
	167	58.88	71.90	3
0,989	133	59.35	71.67	1
	134	59.80	72.21	2
	133	59.35	71.67	3
1,094	106	58.40	69.89	1
	108	59.50	71.21	2
	108	59.50	71.21	3
1,217	87	60.12	70.99	1
	87	60.12	70.99	2
	87	60.12	70.99	3

Outro aspecto que se repetiu foi, quando se comparando com a ranhura retangular, a menor divergência entre os valores de preenchimento total obtidos para esta geometria. Novamente houve uma leve redução, e isto reforça o que foi interpretado com a comparação entre os perfis curtos.

Um ponto que fica claro com a implementação do algoritmo na ranhura trapezoidal, de forma geral, é que sua geometria torna mais sutil a diferença causada pela variação da posição do primeiro condutor. Isso torna os resultados mais consistentes e factíveis, devido ao fato de que esta posição é um parâmetro empírico, utilizado somente pela necessidade de estabelecer um ponto de partida para o algoritmo.

Foi obtido um máximo valor de preenchimento, nessa condição, que ocupa 72,71% da área útil ranhura para o condutor de diâmetro nominal 0,71 mm ($d_{isol} = 0,789\ mm$) partindo da posição 1. Ambas as posições 2 e 3 resultaram num menor fator, de 72,02%, e com uma redução de 2 fios.

Os posicionamentos com maior enchimento calculado podem ser vistos na Figura 21a. Para comparação, é apresentado na Figura 21b os posicionamentos que resultaram no menor fator para o mesmo condutor.



(a) 212 condutores, Fator: 72,71%, Diâmetro: 0,789 mm, Posição: 1

(b) 210 condutores, Fator: 72,02%, Diâmetro: 0,789 mm, Posição: 2

Figura 21 – Resultados do algoritmo para a ranhura trapezoidal longa
Fonte: Autoria Própria

4.4 ANÁLISE DA RANHURA PADRÃO

A última das três geometrias analisadas, denominada de ranhura padrão, é o modelo utilizado para a implementação do algoritmo original (RAABE, 2014).

A geometria desta ranhura possui um aspecto mais arredondado, o que é uma característica desenvolvida com o intuito de mesclar a vantagem de um modelo semi-fechado com uma forma de garantir um melhor assentamento dos condutores no seu interior. Por ser utilizada principalmente na bobinagem de máquinas de baixa tensão, essa característica é a mais adequada para a utilização de fios circulares.

4.4.1 Ranhura Padrão de Perfil Curto

Os resultados obtidos para a ranhura padrão são descritos na Tabela 8.

O que pode se concluir a partir dos dados retornados pelo algoritmo, é que os valores mantém-se na mesma consistência enxergada na implementação referente a ranhura trapezoidal. A característica principal que se sobressai neste caso, e que é a principal vantagem da ranhura padrão, é a maior área obtida em relação às suas dimensões externas, o que permite a inserção de um maior número de condutores mantendo um fator de preenchimento elevado.

Novamente é notada uma pequena variação quando se muda a posição do primeiro condutor. Excepcionalmente para o condutor de 0.8 mm, há a maior disparidade entre os valores obtidos. Para este caso, a quantidade de condutores obtido para o

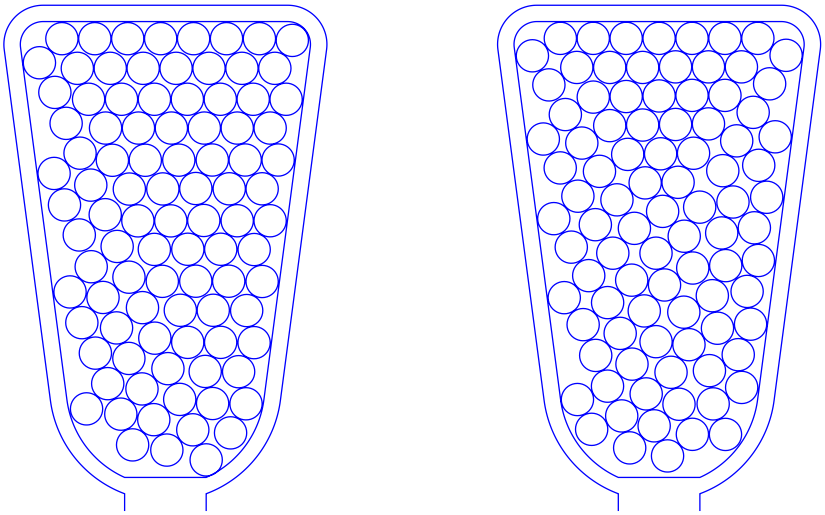
Tabela 8 – Padrão curta

Diâmetro dos condutores mm	Número de condutores	Enchimento cobre %	Enchimento total %	Posição do primeiro condutor
0,789	145	57,95	71,56	1
	144	57,55	71,07	2
	145	57,95	71,56	3
0,884	113	57,33	70,01	1
	117	59,36	72,48	2
	116	58,86	71,86	3
0,989	92	59,08	71,34	1
	94	60,36	72,89	2
	93	59,72	72,11	3
1,094	74	58,66	70,21	1
	75	59,46	71,16	2
	74	58,66	70,21	3
1,217	61	60,66	71,62	1
	61	60,66	71,62	2
	60	59,67	70,45	3

máximo fator na posição 1 é de 113 que, sendo um número primo, torna mais restritas as possíveis tipologias para os enrolamentos (RAABE, 2014).

O maior fator de preenchimento possível de se alcançar nessa ranhura foi calculado em 72,89%, para o condutor de 0,9 mm iniciando na posição 2. Em contrapartida, quando foi alternada para a posição 1, o enchimento caiu para 71,34%, com uma redução de 2 fios.

Essa configuração pode ser visualizada na Figura 22a, que apresenta o caso de maior fator encontrado, e na Figura 22b, que demonstra o caso onde houve maior redução pela alteração da posição inicial.



(a) 94 condutores, Fator: 72,89%, Diâmetro: 0,989 mm, Posição: 2

(b) 92 condutores, Fator: 71,34%, Diâmetro: 0,989 mm, Posição: 1

Figura 22 – Resultados do algoritmo para a ranhura padrão curta
Fonte: Autoria Própria

4.4.2 Ranhura Padrão de Perfil Longo

Os resultados obtidos na implementação configurada para essa ranhura foram dispostos na Tabela 9.

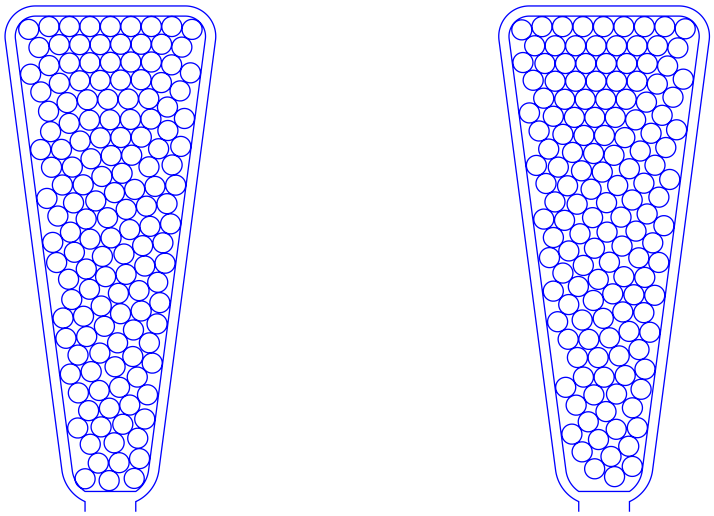
Tabela 9 – Padrão longa

Diâmetro dos condutores mm	Número de condutores	Enchimento cobre %	Enchimento total %	Posição do primeiro condutor
0,789	227	58,69	72,48	1
	229	59,21	73,12	2
	228	58,95	72,80	3
0,884	182	59,74	72,95	1
	181	59,42	72,55	2
	178	58,43	71,34	3
0,989	146	60,66	73,25	1
	144	59,83	72,24	2
	143	59,41	71,74	3
1,094	118	60,52	72,44	1
	118	60,52	72,44	2
	117	60,01	71,82	3
1,217	94	60,48	71,41	1
	94	60,48	71,41	2
	93	59,84	70,65	3

Os dados obtidos para esta configuração de ranhura reforçam a característica principal dessa geometria.

A melhor adequação dos condutores no seu interior tornou possível um leve aumento do fator de preenchimento até mesmo em relação com a ranhura padrão de perfil curto. Essa melhor adequação dos condutores é reiterada como uma vantagem mais evidentemente a partir do que se obteve em relação à disparidade dos resultados para diferentes posições iniciais. Essa característica, presente nos casos com condutores de menor seção transversal, ocorre de forma inversa ao observado na ranhura trapezoidal, onde era mais perceptível a disparidade em condutores maiores.

Para essa implementação, o maior fator de preenchimento de ranhura calculado foi de 73,25%, obtido para o condutor de diâmetro 0,9 mm, a partir da posição inicial 1. Essa configuração pode ser visualizada na Figura 23a. Variando para a posição 3, ocorre uma queda no fator para 71,74%, com uma redução de 3 fios, possível de se observar na Figura 23b.



(a) 146 condutores, Fator: 73,25%, Diâmetro: 0,989 mm, Posição: 1

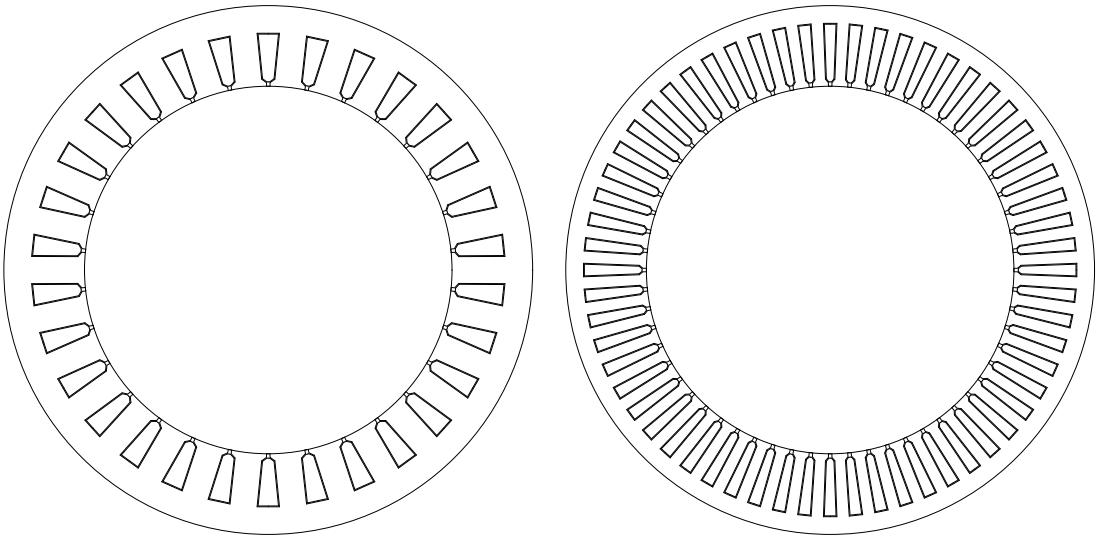
(b) 143 condutores, Fator: 71,74%, Diâmetro: 0,989 mm, Posição: 3

Figura 23 – Resultados do algoritmo para a ranhura padrão longa
Fonte: Autoria Própria

4.5 COMPARAÇÃO DE FATORES DE PREENCHIMENTO DE RANHURAS EM MÁQUINAS DE DIFERENTE NÚMERO DE POLOS

Para determinar a influência dos parâmetros construtivos sobre o fator de preenchimento de ranhura, foi realizada a implementação do algoritmo para outras duas condições.

As ranhuras analisadas pertencem a dois Geradores Síncronos com Imãs Permanentes, G1 e G2, de 10 e 20 polos, respectivamente. Ambas máquinas possuem 3 kVA de potência, tensão nominal de 220 V, frequência de 60 Hz. O rendimento e número de ranhuras por polo por fase são idênticos, bem como o diâmetro externo do estator, de 300 mm. O corte transversal dessas máquinas pode ser visualizado na Figura 24.



(a) G1: Gerador de 10 polos **(b) G2: Gerador de 20 polos**

Figura 24 – Corte transversal dos geradores analisados
Fonte: Autoria própria

A diferença no número de polos causa uma diferença entre as dimensões das ranhuras nos estatores dos geradores, por causa da quantidade de ranhuras por polo por fase ter sido mantida unitária nos dois casos.

A espessura da folha isolante na ranhura e a distância mínima entre objetos também foram consideradas as mesmas, semelhante ao utilizado na Seção 4.1 (0,5 mm e 0,02 mm, respectivamente). Além disso, o diâmetro dos condutores utilizados são próximos entre si.

Os parâmetros dimensionais das ranhuras, bem como dos condutores utilizados, são apresentados na Tabela 10.

Tabela 10 – Parâmetros das ranhuras e condutores utilizados

Parâmetro	Unidade	Variável	G1	G2
Largura	mm	w	12,6914	7,6817
Profundidade	mm	h	30,0893	35,7072
Abertura	mm	w_h	2,00	2,00
Altura da abertura	mm	h_h	2,00	2,00
Altura do colarinho	mm	h_t	2,00	2,00
Ângulo de abertura	$Graus$	α	11,398	4,2596
Área	mm^2	A	272,668	213,501
Área útil	mm^2	A_{util}	236,648	175,255
Diâmetro dos condutores	mm	D_{conds}	1,6481	1,7613

A implementação do algoritmo para ambas as máquinas é análoga aos casos visualizados anteriormente na Seção 4.3. Nesta análise, é estudado o impacto de um parâmetro construtivo, como o número de ranhuras e polos, no fator de preenchimento.

Os resultados referentes a esta implementação são descritos na Tabela 11.

Tabela 11 – Resultados obtidos para os geradores

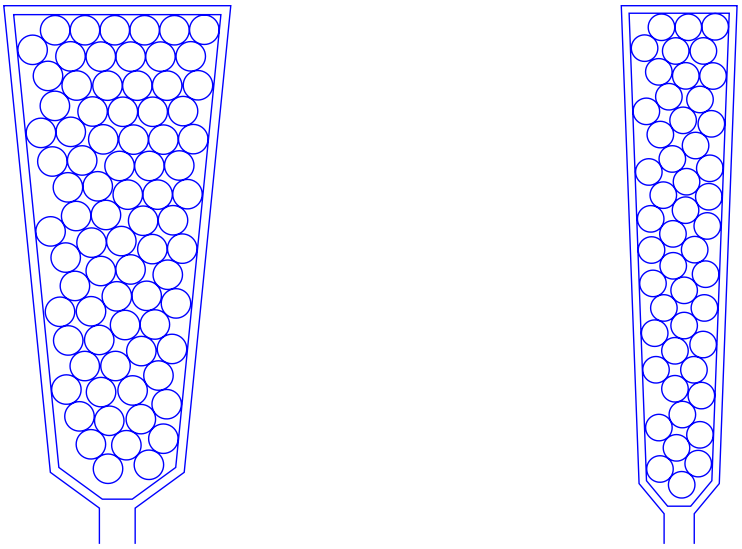
Gerador	Número de condutores	Enchimento total %	Posição
G1	79	71,217	1
	79	71,217	2
	79	71,217	3
G2	48	66,731	1
	49	68,121	2
	49	68,121	3

Os resultados referentes a este estudo possuem um dado que não foi contabilizado. Devido à falta de uma informação referente ao diâmetro dos condutores, não é possível determinar a quantidade de verniz isolante que os envolve. Portanto, o fator de enchimento possível de ser calculado é referente ao diâmetro nu dos condutores. Com isto em mente, o enchimento total e o enchimento de cobre são considerados idênticos, para fim de comparação com os casos anteriores.

Em relação aos valores obtidos, os fatores de preenchimento máximos possuem uma divergência significativa. O maior valor calculado para a ranhura do gerador G1 foi de 71,217% com 79 condutores, nas três posições. Já para o gerador G2 obteve-se 68,121% de preenchimento com 49 condutores. A redução no número de condutores possíveis de serem inseridos na ranhura era esperado, visto que há também uma considerável diminuição na área da ranhura.

Se tratando do fator de preenchimento, a diminuição é resultante do aumento significativo no número de ranhuras, de 30 para 60. Esse aumento, combinado com a

utilização de um estator de mesmas dimensões, reduziu significativamente a largura da ranhura. Outro parâmetro que contribuiu para a diminuição no fator de preenchimento foi o aumento, ainda que pouco significativo, do diâmetro dos condutores. Isso, em conjunto com a redução na largura, impactou na forma com que os mesmos se assentam no interior da ranhura, não permitindo, a partir das condições impostas pela implementação (como posição inicial e sentido de inserção dos condutores), com que sejam posicionados efetivamente de forma hexagonal. O posicionamento dos condutores em ambos os casos de máximo fator de preenchimento calculado, para G1 e G2, podem ser vistos na Figura 25.



(a) G1: 79 condutores, Fator: 71,217%, Posição: 2 **(b) G2: 49 condutores, Fator: 68,121%, Posição: 2**

Figura 25 – Máximos fatores de preenchimentos encontrados para os geradores
Fonte: Autoria própria

A forma hexagonal de posicionamento é possível de ser visualizada na ranhura de G1, porém, em G2, a distribuição sofre uma quebra no padrão a partir da terceira linha de condutores, tendo em vista que a largura comprime de tal forma que já não é mais possível inserir três condutores por linha.

4.6 COMPARAÇÃO COM OS RESULTADOS DOS ARTIGOS TÉCNICOS

Para estabelecer um critério de referência, foi feita uma comparação entre os resultados obtidos pela implementação descrita nos três principais artigos técnicos utilizados como base para o desenvolvimento deste trabalho. As condições analisadas por cada uma das bibliografias estudadas são descritas na sequência.

No artigo de Raabe (2014), é feita a simulação baseada numa ranhura com as dimensões semelhantes à ranhura padrão de perfil curto descrita neste trabalho. A simulação foi realizada para um número de até 4 variações, denominadas *shifts*, no posicionamento do primeiro condutor e com bitolas na faixa entre 0,8 a 1,25 mm;

As mesmas condições de separação mínima entre condutores e de espessura do material isolante de fundo de ranhura foram consideradas.

Os resultados do algoritmo desenvolvido por Raabe (2014) para o condutor de 0,8 mm determinaram um máximo de 115 condutores no interior da ranhura. Em contrapartida, para os mesmos parâmetros, descritos na Seção 4.4.1, o número de condutores máximo obtido para esta configuração foi de 117. Um aumento de 2 condutores que foi possível através do ajuste na sequência em que são inseridos a partir do *loop* realizado pelo algoritmo. Raabe (2014) não especifica a forma com que definiu essa sequência, portanto é difícil comparar a eficiência determinada por essa característica.

Outra divergência encontrada nos resultados foi a determinação da área da ranhura, constatada a partir da comparação entre o fator de preenchimento para o condutor de 0,8 citado anteriormente. No trabalho de Raabe (2014) o preenchimento total para 115 condutores foi calculado como sendo de 60,18%. Isso foi causado pelo cálculo de Raabe ter sido realizado levando em conta a área total da ranhura, que foi determinada por uma aproximação como sendo de 117,3[1] mm^2 (em contraste com o encontrado na Seção 4.1, de 120 mm^2). O enchimento encontrado para 117 condutores, descrito na Seção 4.4.1, foi encontrado a partir da área útil da ranhura, calculada como 99,07 mm^2.

Posteriormente, no desenvolvimento por Tommaso et al. (2017), a análise é expandida para incluir uma ranhura de formato retangular, bem como a inclusão da modelagem com condutores de seção transversal retangular.

Apesar do dimensionamento da ranhura de formato retangular de Tommaso et al. (2017) ser diferente do utilizado na Seção 4.2.1, a comparação será realizada em relação ao formato do agrupamento interno dos condutores. O caso ocorrido neste artigo foi que a inserção dos condutores era separada em *grids* que eram de dimensão determinada pelas características dos fios em combinação com os parâmetros da ranhura. Essa metodologia resultou numa organização de formato hexagonal, ou de colmeia, dos condutores, determinada como sendo uma das geometrias mais otimizadas para o posicionamento de círculos no interior de um retângulo (TOMMASO et al., 2017).

[1] Esse valor foi inferido a partir do cálculo inverso entre o fator de enchimento e a seção transversal dos condutores.

Finalmente, o trabalho mais recente, de Caruso et al. (2018), reutiliza o algoritmo para a análise da implementação em ranhuras retangular e padrão de perfil alongado, utilizadas como base para as descritas nas Seções 4.2.2 e 4.4.2, respectivamente. Este artigo realiza uma modificação na modelagem do condutor de seção retangular e apresenta uma nova metodologia para o algoritmo de posicionamento, tendo como base a inserção em paralelo às bordas inferiores da ranhura, não foi possível realizar comparações assertivas devido a forma com que são apresentados os dados deste artigo.

4.7 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os resultados referentes às implementações do algoritmo para os modelos estudados, em duas variações dimensionais para cada um dos casos. Essa variação teve como intuito demonstrar a versatilidade do algoritmo desenvolvido, que efetua os cálculos e gera as visualizações para qualquer permutação dos parâmetros de entrada descritos na modelagem.

Em resumo, para realizar a comparação dos resultados de cada um dos casos estudados, os valores de enchimentos total e de cobre foram analisados, tendo a média geral e seu desvio padrão descritos na Tabela 12.

Tabela 12 – Comparação dos resultados

Ranhura		Enchimento Cobre		Enchimento Total	
		Média	Desvio Padrão	Média	Desvio Padrão
Curta	Padrão	59.06	1.07	71.34	0.85
	Trapezoidal	58.98	1.08	71.24	1.02
	Retangular	60.28	1.25	72.82	1.13
Longa	Padrão	59.75	0.72	72.17	0.75
	Trapezoidal	59.21	0.69	71.52	0.71
	Retangular	60.82	1.17	73.47	1.38

A partir dos dados expostos, é possível confirmar que o formato geométrico da ranhura foi o maior responsável pela variação no fator de preenchimento. A ranhura retangular foi a que apresentou a maior média geral, porém, devido a um alto desvio padrão, infere-se que a consistência desses valores é grandemente afetada pela posição do primeiro condutor.

Ao mudar para a tipologia semi-fechada das ranhuras trapezoidal e padrão, o arredondamento dos vértices, observado na ranhura padrão, causa uma leve melhora no fator de preenchimento, o que poderá incrementalmente garantir a otimização de um projeto.

Como o dimensionamento da ranhura é fortemente afetado por parâmetros de desempenho da máquina, o fator de enchimento máximo possível também será decorrente dessas condições.

Segundo Hendershot e Miller (2010), o número de ranhuras do estator e a quantidade de polos devem ser definidos em conjunto, tendo em vista que a relação de ranhuras por par de polos afeta não somente na distribuição dos enrolamentos do estator, como também na formação de harmônicos espaciais e na densidade de corrente por condutor. Portanto, a decisão de se manter o valor de ranhuras por polo por fase, aumentando o número de polos, irá afetar de forma inversamente proporcional a largura das ranhuras. Tendo em vista que as dimensões da ranhura irão impactar de forma significativa a maneira com que os condutores se assentam no interior da ranhura, conforme possível observar na comparação realizada na Seção 4.5, afere-se então a consequente influência do número de polos no fator de preenchimento de ranhura.

A metodologia utilizada para o desenvolvimento deste trabalho, descrita no Capítulo 3, conseguiu alcançar resultados incrementalmente mais otimizados quando comparada com as tratativas implementadas nos principais artigos da bibliografia em que foi baseada. Dessa forma, é possível afirmar que este estudo consegue complementar positivamente a aplicabilidade e eficiência da implementação desenvolvida.

5 CONCLUSÕES

O trabalho desenvolvido apresentou a descrição de uma metodologia para a implementação de um algoritmo que realiza o cálculo do fator máximo de preenchimento de ranhura para as geometrias predeterminadas.

A possibilidade de variar os parâmetros dimensionais tanto da ranhura como dos condutores e itens adicionais (distância mínima e folha isolante) incrementa a versatilidade do trabalho e é uma característica chave na busca pela automatização do processo descrito.

A utilização de um algoritmo para calcular o fator máximo de preenchimento para uma geometria predeterminada é uma metodologia que garante diversas vantagens quando aplicada de forma eficaz ao desenvolvimento de um projeto otimizado de uma máquina elétrica.

Com essa ferramenta, que disponibiliza de forma rápida e eficaz a possibilidade de se obter uma visualização da melhor distribuição dos condutores no interior da ranhura, torna-se mais palpável a execução de um projeto que irá atender critérios mais rígidos.

Uma das consequências da determinação desse fator de forma correta e planejada corresponderá na melhora do aproveitamento espacial interno das ranhuras do estator. Isso acarreta tanto em redução de custos, por estar lidando com a quantidade de material ativo, como em uma redução da elevação de temperatura da máquina, que irá permitir uma maior eficiência térmica.

5.1 TRABALHOS FUTUROS

Durante o desenvolvimento, foram identificados alguns assuntos que não eram abrangidos pelo escopo atual do trabalho. Eles podem ser explorados de forma minuciosa futuramente para aperfeiçoar ainda mais a aplicabilidade do algoritmo e garantir a qualidade de seus resultados, os principais identificados são:

- Realizar uma análise comparativa do modelo térmico de uma máquina com fator de preenchimento baixo em contraste com uma de fator elevado, para determinar a real influência deste fator na redução da elevação de temperatura;
- Realizar o comparativo do máximo valor obtido pelo algoritmo do que é possível de ser implementado fisicamente, identificando as principais dificuldades no processo de bobinagem e correlacionando com o cálculo do fator de preenchimento;

- Implementar o algoritmo a uma rotina de otimização dos parâmetros construtivos de máquinas elétricas que ao realizar o cálculo das dimensões da ranhura, sejam determinados também os fatores máximos de preenchimento de ranhura para qualquer variação obtida.

REFERÊNCIAS

CARUSO, M. et al. Algorithmic approach for slot filling factors determination in electrical machines. In: . [S.l.: s.n.], 2018. p. 1489–1494.

HENDERSHOT, J. R.; MILLER, T. J. E. *Design of brushless permanent-magnet machines*. [S.l.]: Motor Design Books, 2010.

IEC. *60317-0-1, Specifications for particular types of winding wires – Part 0-1: General requirements – Enamelled round copper wire*. Geneva, Switzerland, 2013.

JACK, A. G. et al. Permanent-magnet machines with powdered iron cores and prepressed windings. *IEEE Transactions on Industry Applications*, v. 36, n. 4, p. 1077–1084, July 2000. ISSN 0093-9994.

JAKSIC, D. “getting rid of the air”, or how to maximize winding fill factor (id 81). In: *2011 1st International Electric Drives Production Conference*. [S.l.: s.n.], 2011. p. 84–87.

KOLZER, J. F. *Projeto ótimo multidisciplinar de geradores síncronos com imãs permanentes de ferrite para microgeração eólica*. Tese de doutorado em engenharia elétrica — Universidade Federal de Santa Catarina, Centro Tecnológico, Florianópolis, 2017.

MARTELLI, A. *Python in a Nutshell*. O’Reilly Media, Incorporated, 2006. (In a Nutshell (o’Reilly) Series). ISBN 9780596100469. Disponível em: <https://books.google.com.br/books?id=pjqbAgAAQBAJ>.

RAABE, N. An algorithm for the filling factor calculation of electrical machines standard slots. In: IEEE. *Electrical Machines (ICEM), 2014 International Conference on*. [S.l.], 2014. p. 981–986.

TOMMASO, A. D. et al. Fast procedure for the calculation of maximum slot filling factors in electrical machines. In: IEEE. *Ecological Vehicles and Renewable Energies (EVER), 2017 Twelfth International Conference on*. [S.l.], 2017. p. 1–8.

WINTERLE, P. *Vetores e geometria analítica*. Makron Books, 2000. ISBN 9788543002392. Disponível em: <https://books.google.com.br/books?id=AKhivgAACAAJ>.

APÊNDICE A – *SCRIPTS* CONTIDOS NO ALGORITMO

A.1 ORGANIZAÇÃO DO ALGORITMO

O algoritmo está dividido em 6 arquivos, denominados de Módulos. Destes, 5 contém as definições de classes, métodos e características e um deles tem como objetivo realizar a execução principal.

Os módulos que compõem o algoritmo implementado são descritos na sequência:

1. **configuracoes.py**: Arquivo que contém os dados de menor distância entre objetos e do isolante da ranhura;
2. **suporte.py**: Contém a descrição dos métodos de apoio, para utilização no algoritmo, são descritos os cálculos de enchimento, ângulos entre retas, área de triângulos, definição do perímetro para inserção de condutores e das posições iniciais dos primeiros condutores e o método para auxiliar no desenho dos condutores no *Gmsh*;
3. **ranhura.py**: É onde está descrita a Classe Ranhura, que determina os métodos e modelagens das ranhuras;
4. **condutores.py**: Define a Classe Wire, que determina os parâmetros e métodos disponíveis para os condutores. Métodos para o cálculo da distância entre condutores e para determinar se os mesmos estão dentro da ranhura e não se sobrepõem a nenhum outro condutor já inserido;
5. **slot_fill.py**: É onde está descrita a sequência de execução do algoritmo e o código para gerar os arquivos do *Gmsh*;
6. **principal.py**: Utilizado como um ponto central para a execução do código.

A Figura 26 demonstra os módulos e como eles se relacionam.

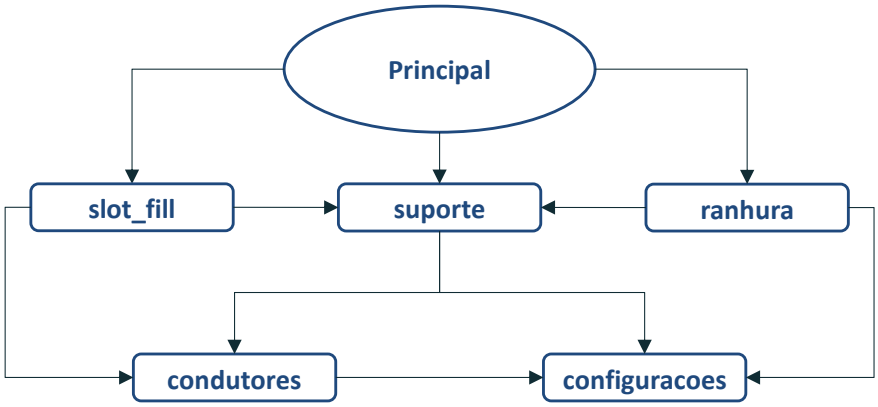


Figura 26 – Organograma do Algoritmo
Fonte: Autoria Própria

A.2 MÓDULO CONFIGURAÇÕES

Determina as variáveis referentes à espessura do isolante da ranhura e ao menor valor possível para a distância entre dois condutores, ou entre um condutor e as bordas da ranhura, conforme pode ser visto no Código A.1.

A decisão de criar um arquivo contendo somente esses dois valores foi pela necessidade de centralização, pois vários dos módulos utilizam essa informação. Caso fossem reescritos em cada um dos arquivos, seria causada duplicidade na informação, tornando mais problemática a alteração desses fatores.

Código A.1 – configuracoes.py

```
1 ISOL_RAN = 0.5  # Espessura do isolante de Fechamento da ranhura
2 DIST_MIN = 0.02  # Distancia de seguranca entre bordas
```

A.3 MÓDULO SUPORTE

O módulo `suporte` oferece funções de apoio para a execução do algoritmo. Conforme é possível visualizar no Código A.2, as funções descritas nesse módulo são:

1. `calcular_enchimento()`: A partir da quantidade de fios, dos dados do isolante da ranhura e do diâmetro dos condutores, calcula o fator de enchimento da ranhura, a partir da área útil (desconsidera o isolante e a área sob a abertura);
2. `angulo_entre_retas()`: Determina o ângulo entre duas retas: A, contendo os pontos `pa` e `pb`, e B, contendo os pontos `pc` e `pd`;
3. `area_triangulo()`: Calcula a área de um triângulo entre os pontos `a`, `b` e `c`;

4. `pontos_circunf()`: Gera `n` pontos (por padrão 360), num raio `r` em volta das coordenadas do `fio_anterior`(por padrão centrado na origem);

5. `draw_wires()`: Gera, a partir de uma lista de condutores, uma *string* formatada para ser inserida no arquivo `.geo` do *Gmsh*;

6. `condutor_inicial()`: Cria o objeto referente ao primeiro condutor `w_0`, a partir dos dados de diâmetro, nu e isolado, da ranhura e da posição inicial;

Código A.2 – suporte.py

```
import numpy as np

from condutores import Wire
from configuracoes import ISOL_RAN, DIST_MIN


def calcular_enchimento(qtd_fios, isol_ranhura, diam):
    secao_fios = qtd_fios * (np.pi * diam**2 / 4)
    enchimento = secao_fios / isol_ranhura.area
    return enchimento


def angulo_entre_retas(pa, pb, pc, pd):
    x1, x2, x3, x4 = pa[0], pb[0], pc[0], pd[0]
    y1, y2, y3, y4 = pa[1], pb[1], pc[1], pd[1]
    dx1 = x2 - x1
    dy1 = y2 - y1
    dx2 = x4 - x3
    dy2 = y4 - y3
    d = dx1*dx2 + dy1*dy2
    l2 = (dx1*dx1+dy1*dy1)*(dx2*dx2+dy2*dy2)
    angle = np.arccos(d/np.sqrt(l2))
    return angle


def area_triangulo(a, b, c):
    area = abs(a[0]*(b[1] - c[1]) + b[0]*(c[1]-a[1]) + c[0]*(a[1]-b[1]))/2
    return area


def pontos_circunf(r, fio_anterior=(0, 0), n=360):
    pi = np.pi
    x_c = fio_anterior[0]
    y_c = fio_anterior[1]
    n = np.radians(n)
    step = n/360
    coordenadas = np.concatenate((np.flip(np.arange(0, pi+step, step), axis=0),
                                  np.arange(pi+step, 2*pi, step)))
    points = [(x_c, y_c+r), (x_c, y_c-r)]
    points = [(np.cos(2*pi/n*x)*r + x_c,  #
               np.sin(2*pi/n*x)*r + y_c) for x in coordenadas]
    return points

```

```
44
45 def draw_wires(wires, r, start=0):
46     n = start
47     str_out = ''
48     for wire in wires:
49         x = wire.x
50         y = wire.y
51         str_out += 'Circle(%s) = {%s, %s, 0, %s, 0, 2*Pi};\n' % (n, x, y, r)
52         n += 1
53     return str_out
54
55
56 def condutor_inicial(diam, d_sem_isol, ranhura, posicao=1):
57     dist = DIST_MIN + ISOL_RAN + (diam/2)
58     if posicao == 1:
59         X_0 = dist
60         Y_0 = 0
61
62     elif posicao == 2:
63         if ranhura.tipo == 1:
64             DIST_1 = ranhura.r1 - dist
65             ANGLE = np.radians(131.5)
66             X_0 = ranhura.BC_c[0] - abs(DIST_1*np.cos(ANGLE))
67             Y_0 = ranhura.BC_c[1] + DIST_1*np.sin(ANGLE)
68
69         elif ranhura.tipo == 2:
70             X_0 = dist
71             Y_VAR = ((0.5/(np.sin(np.radians(90 - ranhura.theta)))) +
72                      DIST_MIN + (diam/2) +
73                      (X_0*np.tan(np.radians(ranhura.theta))))
74             Y_0 = ranhura.B[1] - Y_VAR
75         elif ranhura.tipo == 3:
76             X_0 = dist
77             Y_0 = ranhura.B[1] - dist
78     elif posicao == 3:
79         if ranhura.tipo == 1:
80             DIST_1 = ranhura.r1 - dist
81             ANGLE = np.radians(131.5)
82             X_0 = ranhura.HI_c[0] - abs(DIST_1*np.cos(ANGLE))
83             Y_0 = ranhura.HI_c[1] - DIST_1*np.sin(ANGLE)
84         elif ranhura.tipo == 2:
85             X_0 = dist
86             Y_VAR = ((0.5/(np.sin(np.radians(90 - ranhura.theta)))) +
87                      DIST_MIN + (diam/2) +
88                      (X_0*np.tan(np.radians(ranhura.theta))))
89             Y_0 = ranhura.G[1] + Y_VAR
90         elif ranhura.tipo == 3:
91             X_0 = dist
92             Y_0 = ranhura.F[1] + dist
93
94     w_0 = Wire(X_0, Y_0, diam, d_sem_isol)
95     return w_0
```

A.4 MÓDULO RANHURA

Este módulo contém a descrição da classe Ranhura. Nele estão descritas as modelagens para os três formatos de ranhura (Padrão, Trapezoidal e Retangular), e as variações nos seus perfis (Curto, longo e dos geradores), além disso, é possível informar um parâmetro para alternar entre a definição da ranhura ou do isolante.

Em *Python*, e em linguagens que utilizam programação orientada a objetos, os métodos de uma classe se referem às funções inerentes ao objeto, no caso, denominado de *self*. Portanto, ao ser definida uma função de classe, um dos parâmetros que ela recebe é a referência do próprio objeto. Exemplifica-se esse conceito supondo um objeto da classe Ranhura chamado `ran_1`. Onde houver a utilização do parâmetro `self` na classe, ao ser criado o objeto, deverá ser interpretado como `ran_1`. Assim, a área da ranhura, por exemplo, é referenciada como `ran_1.area`.

O Código A.3 apresenta as definições da classe, que possui os seguintes métodos:

1. `__init__()`: Este é o método padrão utilizado em *Python* para definir um construtor de classe. É a forma com que são definidos os objetos Ranhura. Contém a atribuição dos pontos principais dos três tipos de ranhura e o cálculo de sua área;

2. `draw_slot()`: Gera uma *string* formatada para ser inserida no arquivo `.geo` do *Gmsh*, contendo os pontos e contornos da ranhura;

3. `get_slot()`: Método que contém a modelagem matemática para a definição dos pontos principais da ranhura, a partir das características dimensionais.

   As dimensões das ranhuras utilizadas são informadas no bloco que encontra-se entre as linhas 146 e 196. Sendo assim, para realizar o cálculo em ranhuras com dimensões diferentes das especificadas, deve-se incluir os dados como um novo perfil, dando continuidade aos já existentes, ou, substituindo-os;

Código A.3 – ranhura.py

```
1  import numpy as np
2  import pandas as pd
3
4  from suporte import angulo_entre_retas, area_triangulo
5  from configuracoes import ISOL_RAN
6
7  class Ranhura():
8      '''
9      Classe que constroi os objetos das ranhuras
10     tipo:   1 - Padrao
11             2 - Trapezoidal
```

```
            3 - Retangular

    perfil: 1 - Curto
            2 - Longo
            3 - GI 10 polos
            4 - GI 20 polos

    shift:  0 - Contorno da ranhura
            1 - Isolacao
    '''
    def __init__(self, tipo, perfil, shift=0, dt=pd.DataFrame([])):
        self.tipo = tipo

        if not dt.empty:
            df = dt
            if perfil == 3:
                self.perfil = 'GI_10'
            elif perfil == 4:
                self.perfil = 'GI_20'
        else:
            df = self.get_slot(tipo, perfil, shift)
        self.df = df.copy()
        self.A = (df.at['A', 'X'], df.at['A', 'Y'])
        self.B = (df.at['B', 'X'], df.at['B', 'Y'])
        self.C = (df.at['C', 'X'], df.at['C', 'Y'])
        self.D = (df.at['D', 'X'], df.at['D', 'Y'])
        self.E = (df.at['E', 'X'], df.at['E', 'Y'])
        self.F = (df.at['F', 'X'], df.at['F', 'Y'])
        self.G = (df.at['G', 'X'], df.at['G', 'Y'])
        self.H = (df.at['H', 'X'], df.at['H', 'Y'])

        if tipo == 1:  # Formato da ranhura padrao
            self.formato = 'Padrao'
            self.BC_c = (df.at['BC_c', 'X'], df.at['BC_c', 'Y'])
            self.DE_c = (df.at['DE_c', 'X'], df.at['DE_c', 'Y'])
            self.HI_c = (df.at['HI_c', 'X'], df.at['HI_c', 'Y'])
            self.JK_c = (df.at['JK_c', 'X'], df.at['JK_c', 'Y'])
            self.I_ = (df.at['I', 'X'], df.at['I', 'Y'])
            self.J = (df.at['J', 'X'], df.at['J', 'Y'])
            self.K = (df.at['K', 'X'], df.at['K', 'Y'])
            self.L = (df.at['L', 'X'], df.at['L', 'Y'])
            self.r1 = np.hypot(self.BC_c[0] - self.B[0],
                               self.BC_c[1] - self.B[1])
            r2 = np.hypot(self.DE_c[0] - self.D[0],
                          self.DE_c[1] - self.D[1])
            self.r2 = round(r2, 3)

            self.angle_bc = angulo_entre_retas(self.B, self.BC_c,
                                               self.C, self.BC_c)
            self.angle_de = angulo_entre_retas(self.D, self.DE_c,
                                               self.E, self.DE_c)

            area_arco1 = 0.5 * self.r1**2 * self.angle_bc
            area_arco2 = 0.5 * self.r2**2 * self.angle_de
            area_rect1 = self.r1 * (self.B[1] - self.DE_c[1])
            area_rect2 = self.DE_c[1] * (self.E[0] - self.A[0])
            area_trec1 = ((self.BC_c[1] - self.DE_c[1]) *
```

```
                            (self.DE_c[0] - self.BC_c[0]))/2
            area_trec2 = ((self.E[1] - self.DE_c[1]) *
                            (self.E[0] - self.DE_c[0]))/2
            area_tri1 = area_triangulo(self.C, self.D, self.DE_c)
            area_tri2 = area_triangulo(self.BC_c, self.C, self.DE_c)

            self.area = 2*(area_arco1 + area_arco2 + area_rect1 + area_rect2 +
                           area_trec1 + area_trec2 + area_tri1 + area_tri2)

        elif tipo == 2:  # Formato Trapezoidal
            self.formato = 'Trapezoidal'
            self.I_ = (df.at['I', 'X'], df.at['I', 'Y'])
            self.J = (df.at['J', 'X'], df.at['J', 'Y'])
            area_tri1 = area_triangulo(self.A, self.B, self.C)
            area_tri2 = area_triangulo(self.A, self.C, self.D)
            third_point = (self.D[0], 0)
            area_tri3 = area_triangulo(self.A, self.D, third_point)
            self.area = 2 * (area_tri1 + area_tri2 + area_tri3)
        elif tipo == 3:  # Formato Retangular
            self.formato = 'Retangular'
            self.area = 2 * (self.B[1] - self.A[1]) * (self.C[0] - self.B[0])

        else:
            raise KeyError('Tipo de ranhura nao existente')

    def draw_slot(self, start_point=0, start_obj=0):
        data = self.df
        if self.tipo == 1:
            string_out = ''
            n = start_obj
            for i in data.index:
                px = data.at[i, 'X']
                py = data.at[i, 'Y']
                pt = data.index.get_loc(i) + start_point
                string_out += 'Point(%s) = {%s, %s, 0, 1e-2};\n' % (pt, px, py)
                if i in ['B', 'D', 'F', 'H', 'J', 'L']:
                    str_ = '{}, {}'.format(pt-1, pt)
                    string_out += 'Line(%s) = {%s};\n' % (n, str_)
                    n += 1
                if i in ['C', 'E', 'I', 'K']:
                    str_ = '{}, {}, {}'.format(pt-2, pt-1, pt)
                    string_out += 'Circle(%s) = {%s};\n' % (n, str_)
                    n += 1

        if self.tipo == 2:
            string_out = ''
            n = start_obj
            for i in data.index:
                px = data.at[i, 'X']
                py = data.at[i, 'Y']
                pt = data.index.get_loc(i) + start_point
                string_out += 'Point(%s) = {%s, %s, 0, 1e-2};\n' % (pt, px, py)
                if i not in ['A', 'F']:
                    str_ = '{}, {}'.format(pt-1, pt)
                    string_out += 'Line(%s) = {%s};\n' % (n, str_)
                    n += 1

```

```
126         if self.tipo == 3:
127             string_out = ''
128             n = start_obj
129             for i in data.index:
130                 px = data.at[i, 'X']
131                 py = data.at[i, 'Y']
132                 pt = data.index.get_loc(i) + start_point
133                 string_out += 'Point(%s) = {%s, %s, 0, 1e-2};\n' % (pt, px, py)
134                 if i not in ['A', 'E']:
135                     str_ = '{}, {}'.format(pt-1, pt)
136                     string_out += 'Line(%s) = {%s};\n' % (n, str_)
137                     n += 1
138 
139         return (string_out, n, pt+1)
140 
141     def get_slot(self, tipo, perfil, shift=0):
142         i = shift
143         d = ISOL_RAN
144         id_ = i*d
145         alpha = 15  # alpha padrao
146 # Ranhuras ============================================================================
147 
148     # Ranhuras curtas
149         if perfil == 1:
150             self.perfil = 'Curta'
151             w = 9.88 - 2*id_
152             h = 15.5 - id_
153             w_h = 2.5
154             h_h = 0.53
155             h_t = 2.3351 - id_
156             base_r1 = 1.2
157             base_r2 = 3.54
158             r1 = base_r1 - id_
159             r2 = base_r2 - id_
160 
161     # Ranhuras longas
162         elif perfil == 2:
163             self.perfil = 'Longa'
164             w = 10.36 - 2*id_
165             h = 24.93 - id_
166             w_h = 2.5
167             h_h = 0.5
168             h_t = 2.3351
169             base_r1 = 1.4998
170             base_r2 = 1.9711
171             r1 = base_r1 - id_
172             r2 = base_r2 - id_
173 
174     # Ranhura 1 - Gerador de 10 polos
175         elif perfil == 3:
176             self.perfil = 'GI_10'
177             w = 12.6914
178             h_h = 2
179             h_t = 2
180             h = 26.0893 + h_t + h_h
181             w_h = 2
182             alpha = 2 * 5.6990
```

```
183
184     # Ranhura 2 - Gerador de 20 polos
185         elif perfil == 4:
186             self.perfil = 'GI_20'
187             w = 7.6817
188             h_h = 2
189             h_t = 2
190             h = 31.7072 + h_t + h_h
191             w_h = 2
192             alpha = 2 * 2.1298
193         else:
194             self.perfil = 'Nulo'
195             raise ValueError('Perfil nao configurado')
196 # ============================================================================
197
198         self.theta = alpha/2
199         theta = np.radians(self.theta)
200
201         if tipo == 1:  # Padrao
202             x_A = 0 + id_
203             y_A = 0
204
205             x_B = 0 + id_
206
207             y_C = w / 2
208
209             xc_BC = x_B + r1
210             yc_BC = (w/2) - r1*np.cos(theta)
211
212             x_C = xc_BC + r1 * (np.sin(theta))
213
214             y_B = yc_BC
215
216             x_E = h - h_h
217             y_E = w_h/2
218
219             x_F = h
220             y_F = w_h/2
221
222             if i == 1:
223                 x_F = x_E
224                 y_F = 0
225
226             s_CD = np.tan(-theta)
227             b_CD = y_C - s_CD * x_C
228
229             x1 = r2 * np.sin(theta) + x_E
230             y1 = y_E - b_CD + r2 * np.cos(theta)
231
232             p = -2 * ((s_CD*y1 + x1)/(s_CD**2 + 1))
233             q = (y1**2 + x1**2 - r2**2)/(s_CD**2 + 1)
234
235             aux = ((p/2)**2 - q)
236
237             x_D = (-p/2) - np.sqrt(aux)
238             y_D = s_CD * x_D + b_CD
239
```

```
240             xc_DE = x_D-r2*np.sin(theta)
241             yc_DE = s_CD * x_D + b_CD - r2 * np.cos(theta)
242 
243         #   Criacao do vetor com as coordenadas
244             vecX = [x_A, x_B, xc_BC, x_C, x_D, xc_DE, x_E, x_F]
245             vecY = [y_A, y_B, yc_BC, y_C, y_D, yc_DE, y_E, y_F]
246             vecMY = [-x for x in vecY]
247             vecNames = ['A', 'B', 'BC_c', 'C', 'D', 'DE_c', 'E', 'F']
248             vecMNames = ['G', 'H', 'HI_c', 'I', 'J', 'JK_c', 'K', 'L']
249 
250         if tipo == 2:  # Trapezoidal
251             x_A = 0 + id_
252             y_A = 0
253 
254             x_B = 0 + id_
255             y_B = (w + 2*id_)/2 - (id_/np.cos(theta) + x_B*np.tan(theta))
256 
257             x_D = h - h_h
258             y_D = w_h / 2 - 0.5*id_
259 
260             x_E = h
261             y_E = w_h / 2
262 
263             if i == 1:
264                 x_E = x_D
265                 y_E = 0
266 
267             x_C = h - h_h - h_t
268             s_BC = np.tan(-theta)
269             b_BC = y_B - s_BC * x_B
270             y_C = s_BC * x_C + b_BC
271 
272         #   Criacao do vetor com as coordenadas
273             vecX = [x_A, x_B, x_C, x_D, x_E]
274             vecY = [y_A, y_B, y_C, y_D, y_E]
275             vecMY = [-x for x in vecY]
276             vecNames = ['A', 'B', 'C', 'D', 'E']
277             vecMNames = ['F', 'G', 'H', 'I', 'J']
278 
279         if tipo == 3:  # Retangular
280             x_A = 0 + id_
281             y_A = 0
282 
283             x_B = 0 + id_
284             y_B = w / 2
285 
286             x_C = h - h_h
287             y_C = w / 2
288 
289             x_D = h
290             y_D = w / 2
291 
292             if i == 1:
293                 x_D = x_C
294                 y_D = 0
295 
296             # Cria   o do vetor com as coordenadas
```

```
297             vecX = [x_A, x_B, x_C, x_D]
298             vecY = [y_A, y_B, y_C, y_D]
299             vecMY = [-x for x in vecY]
300             vecNames = ['A', 'B', 'C', 'D']
301             vecMNames = ['E', 'F', 'G', 'H']
302 
303         vecX = vecX + vecX
304         vecY = vecY + vecMY
305         vecNames = vecNames + vecMNames
306         data = {'Ponto': vecNames, 'X': vecX, 'Y': vecY}
307 
308         df = pd.DataFrame(data)
309         df.set_index('Ponto', inplace=True)
310         return df
```

A.5 MÓDULO CONDUTORES

Este módulo descreve as características dos condutores e os métodos que definem a interação destes com a ranhura, ou entre si. O Código A.4 apresenta as definições da classe, que possui os seguintes métodos:

1. `__init__()`: Construtor da classe. Contém a atribuição dos parâmetros de diâmetro, nu e isolado, o ponto central do condutor e a característica que define se o condutor está finalizado, inicializada como falsa;

2. `distant_to_wires()`: Calcula a distância entre o condutor `self` e outro, denominado `wire2`. Retorna verdadeiro se essa distância for maior ou igual ao mínimo permitido, caso contrário retorna falso;

3. `distance_to_line()`: Calcula a distância do condutor até uma reta definida pelos pontos `p1` e `p2`;

4. `distance_to_arc()`: Calcula a distância do condutor até o centro de um arco;

5. `inside_slot()`: Determina se o condutor está dentro da ranhura, através da leitura dos pontos principais de um objeto Ranhura. As análises são feitas para cada tipo de ranhura, e para os setores definidos na Seção 3.4. As verificações são feitas a partir das coordenadas x e y no plano cartesiano, se, e somente se, ambas estiverem marcadas como verdadeiras, é validada a posição do condutor como sendo interna a ranhura;

6. `not_overlapping()`: Recebe a lista de condutores já inseridos na ranhura para determinar se o condutor `self`, a ser posicionado, não irá sobrepôr nenhum;

Código A.4 – condutores.py

```
1
2  import numpy as np
3
4  from configuracoes import DIST_MIN, ISOL_RAN
5
6
7  class Wire():
8      def __init__(self, xc, yc, d, d_sem_isol):
9          self.diam = d
10         self.d_sem_isol = d_sem_isol
11         self.dconds = self.diam + DIST_MIN
12         self.dtotal = (self.diam/2) + ISOL_RAN + DIST_MIN
13         self.x = xc
14         self.y = yc
15         self.pos = (self.x, self.y)
16         self.finalizado = False
17
18     def distant_to_wires(self, wire2):
19
20         distance = np.hypot(self.x - wire2.x, self.y - wire2.y)
21         return bool(distance >= self.dconds)
22
23     def distance_to_line(self, p1, p2):
24         x_diff = p2[0] - p1[0]
25         y_diff = p2[1] - p1[1]
26         num = abs(y_diff*self.x - x_diff*self.y + p2[0]*p1[1] - p2[1]*p1[0])
27         den = np.sqrt(y_diff**2 + x_diff**2)
28         return round(num / den, 4)
29
30     def distance_to_arc(self, arc_center_points):
31         x_arc = arc_center_points[0]
32         y_arc = arc_center_points[1]
33         distance = np.hypot(self.x - x_arc, self.y - y_arc)
34         return distance
35
36     def inside_slot(self, slot):
37         x, y = False, False
38
39         if slot.tipo == 1:
40
41             # Area 1
42             if (slot.A[0] <= self.x <= slot.C[0]):
43                 # Retangulo entre os arcos
44                 into_rect = (slot.H[1] < self.y < slot.B[1])
45                 inside = self.distance_to_line(slot.H, slot.B) >= self.dtotal
46                 if into_rect and inside:
47                     x, y = True, True
48                 else:
49                     # Dentro de um dos arcos
50                     DIST_ARCO = slot.r1 - self.dtotal
51                     if ((self.y >= slot.B[1] and
52                          self.distance_to_arc(slot.BC_c) <= DIST_ARCO) or
53                             (self.y <= slot.H[1] and
54                              self.distance_to_arc(slot.HI_c) <= DIST_ARCO)):
55                         x, y = True, True
```

```
56
57              # Area 2
58              elif slot.C[0] < self.x <= slot.D[0]:
59                  x = True
60                  cima = self.distance_to_line(slot.C, slot.D) >= self.dtotal
61                  baixo = self.distance_to_line(slot.I_, slot.J) >= self.dtotal
62                  if cima and baixo:
63                      y = True
64
65              # Area 3
66              elif (slot.D[0] <= self.x <= slot.E[0]):
67                  # Retangulo entre os arcos
68                  into_rect = (slot.K[1] < self.y < slot.E[1])
69                  inside = self.distance_to_line(slot.K, slot.E) >= self.dtotal
70                  if into_rect and inside:
71                      x, y = True, True
72                  else:
73                      # Dentro de um dos arcos
74                      DIST_ARCO = slot.r2 - self.dtotal
75                      if ((self.y >= slot.E[1] and
76                           self.distance_to_arc(slot.DE_c) <= DIST_ARCO) or
77                          (self.y <= slot.K[1] and
78                           self.distance_to_arc(slot.JK_c) <= DIST_ARCO)):
79                          x, y = True, True
80          if slot.tipo == 2:
81              # Ranhura trapezoidal
82              if slot.A[0] < self.x <= slot.D[0]:
83                  esquerda = self.distance_to_line(slot.G, slot.B) >= self.dtotal
84                  direita = self.distance_to_line(slot.D, slot.I_) >= self.dtotal
85                  if esquerda and direita:
86                      x = True
87
88                  cima_1 = self.distance_to_line(slot.B, slot.C) >= self.dtotal
89                  baixo_1 = self.distance_to_line(slot.G, slot.H) >= self.dtotal
90
91                  cima_2 = self.distance_to_line(slot.C, slot.D) >= self.dtotal
92                  baixo_2 = self.distance_to_line(slot.H, slot.I_) >= self.dtotal
93                  if cima_1 and baixo_1 and cima_2 and baixo_2:
94                      y = True
95
96          if slot.tipo == 3:
97              if slot.A[0] < self.x <= slot.C[0]:
98                  esquerda = self.distance_to_line(slot.F, slot.B) >= self.dtotal
99                  direita = self.distance_to_line(slot.G, slot.C) >= self.dtotal
100                 if esquerda and direita:
101                     x = True
102                 cima = self.distance_to_line(slot.B, slot.C) >= self.dtotal
103                 baixo = self.distance_to_line(slot.F, slot.G) >= self.dtotal
104                 if cima and baixo:
105                     y = True
106
107         return bool(x and y)
108
109     def not_overlapping(self, wires):
110         check = True
111         for wire in wires:
112             distance = self.distant_to_wires(wire)
```

```
113             if distance:
114                 continue
115             else:
116                 return False
117         return check
```

A.6 MÓDULO SLOT_FILL

Este módulo contém a lógica principal por trás do programa, nele são descritos os métodos `slot_fill()`, que executa em *loop* a sequência de operação do algoritmo, descrita na Seção 3.5, e `draw_gmsh()`, que faz a união das informações dos condutores e da ranhura para gerar o arquivo `.geo` do *script* do *Gmsh*, além de armazenar os principais dados da execução numa planilha.

No Código A.5, é possível ver a definição da função `slot_fill()` iniciando na linha 9, os parâmetros de entrada são: `w_0`, que é um objeto da classe *Wire* e contém as informações do primeiro condutor a ser inserido na ranhura; e `ranhura`, um objeto da classe Ranhura, que descreve o contorno da ranhura. Após o término do processo iterativo do algoritmo, ou seja, assim que todos os condutores na ranhura estiverem marcados como finalizados (não possuem pontos no seu perímetro para inserção de novos condutores), a função retorna a lista com os condutores inseridos.

A partir da linha 36 é realizada a definição da função `draw_gmsh()` que recebe os parâmetros com dados da ranhura, do isolante, dos fios e o número referente a posição inicial.

Código A.5 – slot_fill.py

```
1  # Biblioteca externa
2  import pandas as pd
3
4  # Classes definidas para as ranhuras e condutores
5  from condutores import Wire
6  from suporte import calcular_enchimento, pontos_circunf, draw_wires
7
8
9  def slot_fill(w_0, ranhura):
10
11     wires = []
12     diam = w_0.diam
13     d_sem_isol = w_0.d_sem_isol
14     wires.append(w_0)
15
16     for fio in wires:
17         if not fio.finalizado:
18             for def_wire in wires:
19                 possible_positions = pontos_circunf(def_wire.dconds,
20                                                     def_wire.pos)
21                 for point in possible_positions:
22                     new_point = Wire(point[0], point[1], diam, d_sem_isol)
```

```
                    inside = new_point.inside_slot(ranhura)
                    not_overlapping = new_point.not_overlapping(wires)
                    if inside and not_overlapping:
                        wires.append(new_point)
                    else:
                        continue
                def_wire.finalizado = True
        else:
            continue

    return wires


def draw_gmsh(ranhura, isol_ranhura, wires, posicao):
    diametro = wires[0].diam
    raio = diametro / 2
    fio = wires[0].d_sem_isol

    STR_START = 'SetFactory("OpenCASCADE");\n'
    OUT = ranhura.draw_slot()
    OUT2 = isol_ranhura.draw_slot(OUT[2], OUT[1])
    OUT3 = draw_wires(wires, raio, OUT2[1])

    str_out = STR_START + OUT[0] + OUT2[0] + OUT3

    ench_total = round(calcular_enchimento(len(wires),
                                           isol_ranhura,
                                           diametro) * 100, 3)

    ench_cobre = round(calcular_enchimento(len(wires),
                                           isol_ranhura,
                                           fio) * 100, 3)

    dict_add = {'Formato': [ranhura.formato],
                'Comprimento': [ranhura.perfil],
                'Posicao': [posicao],
                'Area': [round(ranhura.area, 3)],
                'Area util': [round(isol_ranhura.area, 3)],
                'Numero de condutores': [len(wires)],
                'Diametro dos condutores': [diametro],
                'Enchimento cobre': [ench_cobre],
                'Enchimento total': [ench_total]
                }

    F1 = '.\\%s_%s_fio%s_%s.geo' % (ranhura.formato.lower(),
                                    ranhura.perfil.lower(),
                                    fio*100, posicao)
    with open(F1, 'w') as f:
        f.write(str_out)

    add = pd.DataFrame.from_dict(dict_add)
    return add
```

A.7 MÓDULO PRINCIPAL

Este *script* é a rotina de execução dos métodos do algoritmo. É realizada a importação de três módulos: `ranhura`; `slot_fill` e `suporte`. Este módulo irá variar de acordo com a ranhura a ser preenchida. Desta forma, um exemplo da utilização deste programa pode ser visualizada no Código A.6, onde é feito o procedimento para a ranhura padrão curta, com condutores de 0,8 mm de diâmetro.

Código A.6 – principal.py

```
# Exemplo de execucao do algoritmo

from ranhura import Ranhura
from slot_fill import slot_fill, draw_gmsh
from suporte import condutor_inicial

# Define a ranhura e o isolante
ranhura = Ranhura(tipo=1, perfil=1)  # Tipo: Padrao, Perfil: Curto
isol_ranhura = Ranhura(tipo=1, perfil=1, shift=1)  # shift=1 gera o isolante

# Parametros dos condutores
diam = 0.884
d_sem_isol = 0.8
posicao = 2  # Posicao inicial do primeiro condutor

# Gera o objeto do primeiro condutor
w_0 = condutor_inicial(diam,
                       d_sem_isol,
                       ranhura,
                       posicao)

# Calcula as posicoes dos condutores e salva a lista na variavel wires
wires = slot_fill(w_0, ranhura)

# Gera o arquivo do Gmsh para visualizacao e retorna os dados do calculo
resultados = draw_gmsh(ranhura, isol_ranhura, wires, posicao)
```

O arquivo `.geo` gerado por esse exemplo pode ser visualizado no Código A.7.

Código A.7 – Arquivo de saída do algoritmo

```
SetFactory("OpenCASCADE");
Point(0) = {0.0, 0.0, 0, 1e-2};
Point(1) = {0.0, 3.750266166351428, 0, 1e-2};
Line(0) = {0, 1};
Point(2) = {1.2, 3.750266166351428, 0, 1e-2};
Point(3) = {1.3566314306640619, 4.94, 0, 1e-2};
Circle(1) = {1, 2, 3};
Point(4) = {12.114873364934716, 3.523650579703813, 0, 1e-2};
Line(2) = {3, 4};
Point(5) = {11.652810644475734, 0.013935770440524298, 0, 1e-2};
Point(6) = {14.97, 1.25, 0, 1e-2};
Circle(3) = {4, 5, 6};
Point(7) = {15.5, 1.25, 0, 1e-2};
```

```
Line(4) = {6, 7};
Point(8) = {0.0, 0.0, 0, 1e-2};
Point(9) = {0.0, -3.750266166351428, 0, 1e-2};
Line(5) = {8, 9};
Point(10) = {1.2, -3.750266166351428, 0, 1e-2};
Point(11) = {1.3566314306640619, -4.94, 0, 1e-2};
Circle(6) = {9, 10, 11};
Point(12) = {12.114873364934716, -3.523650579703813, 0, 1e-2};
Line(7) = {11, 12};
Point(13) = {11.652810644475734, -0.013935770440524298, 0, 1e-2};
Point(14) = {14.97, -1.25, 0, 1e-2};
Circle(8) = {12, 13, 14};
Point(15) = {15.5, -1.25, 0, 1e-2};
Line(9) = {14, 15};
Point(16) = {0.5, 0.0, 0, 1e-2};
Point(17) = {0.5, 3.745988597038333, 0, 1e-2};
Line(10) = {16, 17};
Point(18) = {1.2, 3.745988597038333, 0, 1e-2};
Point(19) = {1.291368334554036, 4.44, 0, 1e-2};
Circle(11) = {17, 18, 19};
Point(20) = {12.093961063701284, 3.0178116867883222, 0, 1e-2};
Line(12) = {19, 20};
Point(21) = {11.697161439352328, 0.0038193082119386546, 0, 1e-2};
Point(22) = {14.47, 1.25, 0, 1e-2};
Circle(13) = {20, 21, 22};
Point(23) = {14.47, 0.0, 0, 1e-2};
Line(14) = {22, 23};
Point(24) = {0.5, 0.0, 0, 1e-2};
Point(25) = {0.5, -3.745988597038333, 0, 1e-2};
Line(15) = {24, 25};
Point(26) = {1.2, -3.745988597038333, 0, 1e-2};
Point(27) = {1.291368334554036, -4.44, 0, 1e-2};
Circle(16) = {25, 26, 27};
Point(28) = {12.093961063701284, -3.0178116867883222, 0, 1e-2};
Line(17) = {27, 28};
Point(29) = {11.697161439352328, -0.0038193082119386546, 0, 1e-2};
Point(30) = {14.47, -1.25, 0, 1e-2};
Circle(18) = {28, 29, 30};
Point(31) = {14.47, 0.0, 0, 1e-2};
Line(19) = {30, 31};
Circle(20) = {1.0422964285246543, 3.9285176278992107, 0, 0.442, 0, 2*Pi};
Circle(21) = {0.963507637080756, 3.027957620824274, 0, 0.442, 0, 2*Pi};
Circle(22) = {1.7917463941143974, 3.4230072431616336, 0, 0.442, 0, 2*Pi};
Circle(23) = {0.979284612500044, 2.124095304402896, 0, 0.442, 0, 2*Pi};
Circle(24) = {2.617591487823309, 3.790697168502157, 0, 0.442, 0, 2*Pi};
Circle(25) = {1.8075233695336852, 2.5191449267402555, 0, 0.442, 0, 2*Pi};
Circle(26) = {0.9635076370807236, 1.2202329879815184, 0, 0.442, 0, 2*Pi};
Circle(27) = {2.6333684632425967, 2.8868348520807787, 0, 0.442, 0, 2*Pi};
Circle(28) = {3.4434365815322088, 3.423007243161608, 0, 0.442, 0, 2*Pi};
Circle(29) = {1.7917463941143648, 1.615282610318878, 0, 0.442, 0, 2*Pi};
Circle(30) = {0.9635076370807074, 0.31623298798151833, 0, 0.442, 0, 2*Pi};
Circle(31) = {2.6175914878232764, 1.9829725356594012, 0, 0.442, 0, 2*Pi};
Circle(32) = {3.4162554282637156, 2.4348348520807552, 0, 0.442, 0, 2*Pi};
Circle(33) = {4.340698302615964, 3.5331771295998613, 0, 0.442, 0, 2*Pi};
Circle(34) = {1.7917463941143486, 0.711282610318878, 0, 0.442, 0, 2*Pi};
Circle(35) = {0.9635076370806912, -0.5877670120184817, 0, 0.442, 0, 2*Pi};
Circle(36) = {2.61759148782326, 1.078972535659401, 0, 0.442, 0, 2*Pi};
```

```
Circle(37) = {4.303646402100404, 2.6073261839011517, 0, 0.442, 0, 2*Pi};
Circle(38) = {3.4320324036830034, 1.530972535659377, 0, 0.442, 0, 2*Pi};
Circle(39) = {5.107333781541359, 3.0541301147330215, 0, 0.442, 0, 2*Pi};
Circle(40) = {1.7917463941143323, -0.19271738968112206, 0, 0.442, 0, 2*Pi};
Circle(41) = {0.9792846124999791, -1.4916293284398598, 0, 0.442, 0, 2*Pi};
Circle(42) = {2.633368463242548, 0.17511021923802306, 0, 0.442, 0, 2*Pi};
Circle(43) = {4.319423377519692, 1.7034638674797735, 0, 0.442, 0, 2*Pi};
Circle(44) = {3.432032403682987, 0.626972535659377, 0, 0.442, 0, 2*Pi};
Circle(45) = {5.967088872272178, 3.333481477647974, 0, 0.442, 0, 2*Pi};
Circle(46) = {5.1231107569606475, 2.1502677983116434, 0, 0.442, 0, 2*Pi};
Circle(47) = {1.8075233695336201, -1.0965797061025002, 0, 0.442, 0, 2*Pi};
Circle(48) = {0.995061587919267, -2.395491644861238, 0, 0.442, 0, 2*Pi};
Circle(49) = {2.6491454386618356, -0.728752097183355, 0, 0.442, 0, 2*Pi};
Circle(50) = {4.319423377519676, 0.7994638674797735, 0, 0.442, 0, 2*Pi};
Circle(51) = {3.447809379102275, -0.276889780762001, 0, 0.442, 0, 2*Pi};
Circle(52) = {5.982865847691467, 2.429619161226596, 0, 0.442, 0, 2*Pi};
Circle(53) = {6.772558770138451, 2.9230740658833994, 0, 0.442, 0, 2*Pi};
Circle(54) = {5.107333781541327, 1.246405481890266, 0, 0.442, 0, 2*Pi};
Circle(55) = {1.823300344952908, -2.0004420225238784, 0, 0.442, 0, 2*Pi};
Circle(56) = {1.0108385633385548, -3.299353961282616, 0, 0.442, 0, 2*Pi};
Circle(57) = {2.7279342301057015, -1.6293121042582945, 0, 0.442, 0, 2*Pi};
Circle(58) = {4.335200352938965, -0.10439844894160455, 0, 0.442, 0, 2*Pi};
Circle(59) = {3.5265981705461407, -1.1774497878369405, 0, 0.442, 0, 2*Pi};
Circle(60) = {5.9670888722721465, 1.5257568448052183, 0, 0.442, 0, 2*Pi};
Circle(61) = {6.765752812712586, 1.9776191612265723, 0, 0.442, 0, 2*Pi};
Circle(62) = {7.653389308704304, 3.1264298190102533, 0, 0.442, 0, 2*Pi};
Circle(63) = {5.123110756960616, 0.34254316546888786, 0, 0.442, 0, 2*Pi};
Circle(64) = {1.8390773203721957, -2.9043043389452565, 0, 0.442, 0, 2*Pi};
Circle(65) = {1.6611217428446785, -3.9273251281775674, 0, 0.442, 0, 2*Pi};
Circle(66) = {2.664874377840997, -2.531110005693175, 0, 0.442, 0, 2*Pi};
Circle(67) = {3.5028134699403974, -2.0949065239770066, 0, 0.442, 0, 2*Pi};
Circle(68) = {4.413989144382831, -1.004958456016544, 0, 0.442, 0, 2*Pi};
Circle(69) = {5.982865847691435, 0.6218945283838403, 0, 0.442, 0, 2*Pi};
Circle(70) = {7.638949759677904, 2.211591577999251, 0, 0.442, 0, 2*Pi};
Circle(71) = {6.781529788131874, 1.0737568448051942, 0, 0.442, 0, 2*Pi};
Circle(72) = {8.428268548539, 2.6608353992915412, 0, 0.442, 0, 2*Pi};
Circle(73) = {5.201899548404482, -0.5580168416060517, 0, 0.442, 0, 2*Pi};
Circle(74) = {2.4893604998783196, -3.5322755058402078, 0, 0.442, 0, 2*Pi};
Circle(75) = {3.3151575573471206, -3.159081172588126, 0, 0.442, 0, 2*Pi};
Circle(76) = {4.390204443777086, -1.9224151921566102, 0, 0.442, 0, 2*Pi};
Circle(77) = {6.061654639135301, -0.27866547869109926, 0, 0.442, 0, 2*Pi};
Circle(78) = {7.654726735097192, 1.3077292615778728, 0, 0.442, 0, 2*Pi};
Circle(79) = {6.860318579575741, 0.17319683773025463, 0, 0.442, 0, 2*Pi};
Circle(80) = {9.29724912166724, 2.9100115689501083, 0, 0.442, 0, 2*Pi};
Circle(81) = {8.444045523958287, 1.756973082870163, 0, 0.442, 0, 2*Pi};
Circle(82) = {5.201899548404466, -1.4620168416060517, 0, 0.442, 0, 2*Pi};
Circle(83) = {4.153331761875497, -2.8204368121401417, 0, 0.442, 0, 2*Pi};
Circle(84) = {5.04048762328321, -2.5503863590515614, 0, 0.442, 0, 2*Pi};
Circle(85) = {6.061654639135285, -1.1826654786910993, 0, 0.442, 0, 2*Pi};
Circle(86) = {7.7335155265410584, 0.4071692545029333, 0, 0.442, 0, 2*Pi};
Circle(87) = {6.860318579575725, -0.7308031622697454, 0, 0.442, 0, 2*Pi};
Circle(88) = {9.313026097086528, 2.00614925252873, 0, 0.442, 0, 2*Pi};
Circle(89) = {10.095433745611698, 2.4856092761916386, 0, 0.442, 0, 2*Pi};
Circle(90) = {8.522834315402154, 0.8564130757952235, 0, 0.442, 0, 2*Pi};
Circle(91) = {5.841124078597088, -2.101241371798708, 0, 0.442, 0, 2*Pi};
Circle(92) = {4.722237395384532, -3.522976761297258, 0, 0.442, 0, 2*Pi};
Circle(93) = {5.597045596977587, -3.2627480803120523, 0, 0.442, 0, 2*Pi};
```

```
Circle(94) = {6.700879169327907, -1.8218900088837555, 0, 0.442, 0, 2*Pi};
Circle(95) = {7.7335155265410425, -0.49683074549706674, 0, 0.442, 0, 2*Pi};
Circle(96) = {7.4995431097683465, -1.3700276924624015, 0, 0.442, 0, 2*Pi};
Circle(97) = {9.391814888530394, 1.1055892454537906, 0, 0.442, 0, 2*Pi};
Circle(98) = {10.976264284177551, 2.6889650293184926, 0, 0.442, 0, 2*Pi};
Circle(99) = {10.174222537055565, 1.585049269116699, 0, 0.442, 0, 2*Pi};
Circle(100) = {8.522834315402138, -0.047586924204776504, 0, 0.442, 0, 2*Pi};
Circle(101) = {6.397682052291465, -2.813603093059199, 0, 0.442, 0, 2*Pi};
Circle(102) = {7.257437143022284, -2.5342517301442467, 0, 0.442, 0, 2*Pi};
Circle(103) = {8.372740056733665, -1.1360552756897229, 0, 0.442, 0, 2*Pi};
Circle(104) = {8.08062310892496, -2.0625318690419725, 0, 0.442, 0, 2*Pi};
Circle(105) = {9.391814888530378, 0.20158924545379053, 0, 0.442, 0, 2*Pi};
Circle(106) = {11.055053075621418, 1.788405022243553, 0, 0.442, 0, 2*Pi};
Circle(107) = {11.808400671698546, 2.335744089164163, 0, 0.442, 0, 2*Pi};
Circle(108) = {10.189999512474852, 0.681186952695321, 0, 0.442, 0, 2*Pi};
Circle(109) = {9.16205884559476, -0.6868114543974327, 0, 0.442, 0, 2*Pi};
Circle(110) = {7.997950591059518, -3.0527648286056137, 0, 0.442, 0, 2*Pi};
Circle(111) = {8.95382005589028, -1.828559452269294, 0, 0.442, 0, 2*Pi};
Circle(112) = {10.031039418723001, -0.4376352847388657, 0, 0.442, 0, 2*Pi};
Circle(113) = {11.070830051040705, 0.884542705822175, 0, 0.442, 0, 2*Pi};
Circle(114) = {11.853237699565875, 1.3640027294850836, 0, 0.442, 0, 2*Pi};
Circle(115) = {12.712262988119925, 2.3515210645834674, 0, 0.442, 0, 2*Pi};
Circle(116) = {9.743138844751375, -1.379315630977004, 0, 0.442, 0, 2*Pi};
Circle(117) = {8.84190729661299, -2.728800202216662, 0, 0.442, 0, 2*Pi};
Circle(118) = {9.675786556973017, -2.3726002331987663, 0, 0.442, 0, 2*Pi};
Circle(119) = {10.813926383744134, 0.014364715261134253, 0, 0.442, 0, 2*Pi};
Circle(120) = {10.612119417879615, -1.130139461318437, 0, 0.442, 0, 2*Pi};
Circle(121) = {11.721113230546829, 0.25657153892722373, 0, 0.442, 0, 2*Pi};
Circle(122) = {12.753797706640814, 1.4427915209289666, 0, 0.442, 0, 2*Pi};
Circle(123) = {10.540286056363602, -2.1082962121294124, 0, 0.442, 0, 2*Pi};
Circle(124) = {11.395006382900748, -0.6781394613184369, 0, 0.442, 0, 2*Pi};
Circle(125) = {11.334085918962353, -1.6741802422479093, 0, 0.442, 0, 2*Pi};
Circle(126) = {12.585612729937413, 0.5208755599965778, 0, 0.442, 0, 2*Pi};
Circle(127) = {12.302193229703443, -0.43593263765234747, 0, 0.442, 0, 2*Pi};
Circle(128) = {13.634628245206667, 1.6461472740558205, 0, 0.442, 0, 2*Pi};
Circle(129) = {11.26225255744634, -2.652336993058885, 0, 0.442, 0, 2*Pi};
Circle(130) = {12.198585418352938, -1.4098762211785552, 0, 0.442, 0, 2*Pi};
Circle(131) = {13.458809676902732, 0.7548479767692565, 0, 0.442, 0, 2*Pi};
Circle(132) = {13.166692729094027, -0.17162861658299344, 0, 0.442, 0, 2*Pi};
Circle(133) = {12.126752056836924, -2.388032971989531, 0, 0.442, 0, 2*Pi};
Circle(134) = {13.063084917743522, -1.1455722001092012, 0, 0.442, 0, 2*Pi};
Circle(135) = {13.888659230176765, -0.7156693975124657, 0, 0.442, 0, 2*Pi};
Circle(136) = {12.991251556227509, -2.123728950920177, 0, 0.442, 0, 2*Pi};
```

Quando lido pelo *Gmsh*, o Código A.7 gera o desenho que pode ser visualizado na Figura 27.

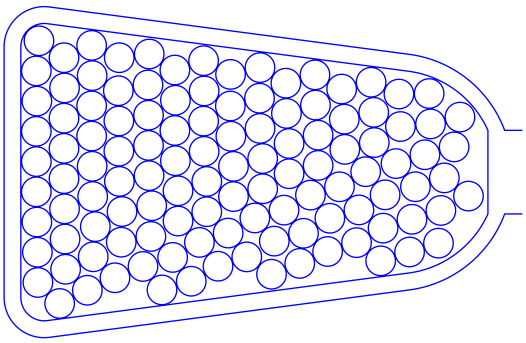


Figura 27 – Ranhura padrão de perfil curto, resultante da leitura do Código A.7 pelo *Gmsh*
Fonte: Autoria Própria