UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
ENGENHARIA ELETRÔNICA E DE TELECOMUNICAÇÕES
CAMPUS PATOS DE MINAS

GUSTAVO JOAQUIM BRITO DOS SANTOS

ESTIMAÇÃO DO COMPRIMENTO DE ANTENAS DE MICROFITA UTILIZANDO *SUPPORT VECTOR REGRESSION*

Patos de Minas - MG

2026

GUSTAVO JOAQUIM BRITO DOS SANTOS

ESTIMAÇÃO DO COMPRIMENTO DE ANTENAS DE MICROFITA UTILIZANDO *SUPPORT VECTOR REGRESSION*

Trabalho de conclusão de curso apresentado à Faculdade de Engenharia Elétrica, da Universidade Federal de Uberlândia, Campus Patos de Minas, Minas Gerais, como requisito parcial à obtenção da graduação em Engenharia Eletrônica e de Telecomunicações.
Orientador: Prof.ª Dr.ª Eliana Pantaleão

Patos de Minas - MG
2026

GUSTAVO JOAQUIM BRITO DOS SANTOS

ESTIMAÇÃO DO COMPRIMENTO DE ANTENAS DE MICROFITA UTILIZANDO *SUPPORT VECTOR REGRESSION*

Trabalho de conclusão de curso apresentado à Faculdade de Engenharia Elétrica, da Universidade Federal de Uberlândia, Campus Patos de Minas, Minas Gerais, como requisito parcial à obtenção da graduação em Engenharia Eletrônica e de Telecomunicações.

Patos de Minas - MG, 4 de setembro de 2026:

Prof.ª Dr.ª Eliana Pantaleão
Orientador

Prof. Dr. Renan Alves do Santos
Coorientador

Prof. Dr. Daniel Costa Ramos – FEELT/UFU
Examinador

Prof. Dr. Pedro Luiz Lima Bertarini – FEELT/UFU
Examinador

Agradecimentos

Dedico esse trabalho a toda minha família, especialmente a minha namorada, Angélica que está sempre ao meu lado e que sempre me impulsiona a concluir a faculdade e a buscar melhores oportunidades e a acreditar em mim mesmo. Meu pai Joaquim, minha mãe Vágma, meu irmão Guilherme, e minha avó Irací, que sempre me apoiaram e supriram em tudo que fosse necessário, de forma que fosse possível me dedicar aos estudos.

Outra parte integral nesse trabalho foi a ajuda de minha orientadora, Eliana, e meu coorientador, Renan, que sempre estiveram dispostos a sanar minhas dúvidas assim que fosse possível, além de me ajudar até mesmo com dúvidas e dicas para uma melhor escrita. Muito obrigado.

Por último, gostaria de agradecer a Universidade Federal de Uberlândia Campus Patos de Minas, por proporcionar a estrutura e o quadro de professores necessária para que os alunos possam ter uma formação excepcional e de forma gratuita. Muito obrigado.

Resumo

Neste trabalho é empregada uma técnica de *Machine Learning* denominada *Support Vector Regression* para regressão de dados de antenas de microfita com geometria retangular para operação em seu modo dominante TM_{10}. Os dados de simulação foram obtidos utilizando o *software Ansys Electronics Desktop* 2022, com o módulo de simulação para alta frequência HFSS. Os resultados das simulações são obtidos em parâmetros S e Z para duas situações diferentes. Primeiramente, variando o comprimento L da antena de 10 mm até 40 mm, com passo de 1mm. Posteriormente, o comportamento da antena é simulado deixando o intervalo de frequência fixo de 1 GHz até 10 GHz, obtendo valores de L que satisfaçam o intervalo estipulado. Ambas as etapas são simuladas para 5 valores diferentes de permissividade do dielétrico ϵ. Com essas informações, é criada uma base de dados cujo objetivo é alimentar modelos de *Support Vector Regression (SVR)*, onde inicialmente é feita limpeza, separação de dados para rotulagem, treinamento e validação. O modelo gerado é capaz de predizer, com eficácia o valor de L para qualquer f_r dentro do intervalo proposto.

Palavras-chave: Antena de microfita. *Machine Learning. Support Vector Regression.* Simulação. Frequência de Ressonância

Abstract

In this paper, a Machine Learning technique called Support Vector Regression is applied for regression of squared microstrip patch antennas data for operation under its dominant mode TM_{10}. The simulation data was gathered through Ansys Electronics Desktop 2022, using the high frequency simulation module HFSS. Simulation results are given in both S and Z parameters for 2 different situations. Firstly, the antenna length L is changed from 10 mm to 40 mm in steps of 1 mm. After it, the antenna behaviour is simulated and the frequency range is fixed from 1 GHz to 10 GHz, getting L values that satisfies the specified frequency range. Both steps are simulated for 5 different dielectric permittivity ϵ. With these informations, a database is created, and it will contain data that will be cleaned, sorted out and labeled, then feeded to a Support Vector Regression (SVR) algorithm for training and validation. The generated model is capable of predicting, accurately, the value of L for any f_r within the frequency range established.

Keywords: Microstrip Patch Antenna. Machine Learning. Support Vector Regression. Simulation. Resonant Frequency

Lista de ilustrações

Lista de códigos

Lista de tabelas

Lista de abreviaturas e siglas

DL	Aprendizado Profundo (*Deep Learning*)
DNN	Rede de Aprendizado Profundo (*Deep Neural Network*)
GMM	Modelos de Mistura de Gaussianas (*Gaussian Mixture Models*)
IA	Inteligência Artificial (*Artificial Inteligence*)
K-Means	Agrupamento K-Means (*K-Means Clustering*)
KKT	Karush Kuhn Tucker (*Karush-Kuhn-Tucker*)
KNN	K-Vizinhos Mais Próximos (*K-Nearest Neighbors*)
ML	Aprendizado de Máquina (*Machine Learning*)
SVM	Máquinas de Vetores de Suporte (*Support Vector Machines*)
SVR	Regressão Por Vetores de Suporte (*Support Vector Regression*)

Lista de símbolos

$f(r, \theta, \phi)$	Função espacial
U	Intensidade de radiação
r	Distância radial
W_{rad}	Densidade de radiação
P_{rad}	Potência irradiada
D	Diretividade
β	Constante de fase do espaço livre
Z	Impedância
Z_A	Impedância da antena
R_τ	Resistência de irradiação
R_L	Resistência de perda da antena
R_A	Resistência da antena
Z_g	Impedância do gerador
R_g	Resistência do gerador
X_g	Reatância do gerador
P_R	Potência de radiação
I_g	Corrente do gerador
P_g	Potência do gerador
x_f	Valor de deslocamento do ponto de ressonância pelo efeito indutivo da alimentação
d	Diametro da sonda de alimentação
h	Espessura do substrato

η	Impedância intrínseca
k	Número de onda
W	Largura do elemento ressonador
L	Comprimento do elemento ressonador
L_p	Comprimento dos cortes casadores de impedância no elemento irradiador
W_p	Largura dos cortes casadores de impedância no elemento irradiador
L_d	Comprimento do dielétrico
W_d	Largura do dielétrico
L_l	Comprimento da linha de alimentação
W_l	Largura da linha de alimentação
ϵ_{ef}	Permissividade elétrica efetiva
μ_0	Permeabilidade elétrica do vácuo
ϵ_0	Permissividade magnética do vácuo
ϵ_r	Permissividade elétrica relativa
G	Condutância de uma antena
$f_{(r)\,mnp}$	Frequência de ressonância para qualquer m,n e p do modo TM
E_ϕ	Campo elétrico de elevação
E_θ	Campo elétrico azimutal
$P(x,y)$	Distribuição de probabilidade
$V(y,f(x))$	Função de perda
H	Espaço de hipóteses
L_{err_min}	Valor mínimo de erro esperado que pode ser alcançado com funções do espaço H

$f(x)$	Função de x
K	*Kernel*
C	Parâmetro de controle de relação de compromisso
ξ_i^-	Variável de restrição de problema inferior
ξ_i^+	Variável de restrição de problema superior
L_ϵ	Função ϵ intensiva
$\alpha \ e \ \alpha^*$	Multiplicadores de Lagrange
R^2	Métrica estatística R^2
y_i	Valor verdadeiro de cada y
$\hat{y}_i$	Valor previsto de cada y
$\bar{y}$	Média dos valores de y
MAE	Métrica MAE
MSE	Métrica MSE
L_{SVR}	Comprimento da antena previsto pelo modelo de SVR
L_{calc}	Comprimento calculado pela equação
$f_{simLSVR}$	Valor de frequência de ressonância obtido pelo *Ansys* utilizando-se L_{SVR}.
$f_{simLcalc}$	Valor de frequência de ressonância obtido pelo *Ansys* utilizando-se L_{calc}.

Sumário

1 Introdução

Na era contemporânea, os computadores foram reduzidos a dispositivos que cabem na palma da mão, e que ainda assim conseguem performar milhões de operações por segundo. O primeiro computador que se assemelha ao contemporâneo é o ENIAC, que usava circuitos eletrônicos e válvulas, além de ocupar o espaço de uma sala inteira e demandar diversas pessoas para programá-lo (1). Um fator que ainda atrasa processos que precisam ser desenvolvidos nessas ferramentas é a necessidade do ser humano em dar instruções ao computador a todo momento, ou ainda que um *software* intermediário seja utilizado para isso, o que nos levou ao desenvolvimento do termo Inteligência Artificial (IA). Esse termo se refere ao desenvolvimento de sistemas que possuem um aspecto de inteligência, semelhante ao de um ser humano, no que tange à capacidade de receber informações a respeito de um problema e descobrir a melhor forma possível de se encontrar a solução de tal problema (2).

Para que informação possa trafegar nos sistemas de telecomunicação, esses sistemas complexos devem pegar um sinal em banda base, modular esse sinal, fazer todo um tratamento de ruído e após isso, transmiti-lo através de um meio físico. Além disso, todo o processo inverso de captar o sinal e levar até um consumidor precisa ser implementado, de tal forma que o circuito final responsável tanto por irradiar quanto receber a informação através de ondas eletromagnéticas é a antena.

Percebe-se então que antenas são extremamente importantes para as telecomunicações e para a vida humana contemporânea no geral. Elas são utilizadas de alguma forma na busca de informações na *internet*, no envio e recebimento de mensagens instantâneas e até mesmo no ato de assistir vídeos no *YouTube*, jogar jogos *online*, assistir TV digital, dentre outros.

Pensando na importância desse componente, pesquisar e aplicar novas formas de otimização de projeto de antenas beneficia não somente a área de telecomunicações em si, mas também impacta toda a sociedade no geral. Pelo fato citado anteriormente, esse projeto busca otimizar o projeto de antenas de microfita, aplicando uma técnica de IA denominada *Support Vector Regression* (SVR).

Este trabalho tem como referência o artigo elaborado em (3) utilizando apenas uma das técnicas apresentadas para a pesquisa, entretanto, utilizando uma base de dados mais significativa, a fim de fazer melhores observações e conclusões gerais a respeito do

modelo de SVR para a aplicação na área de antenas.

1.1 Tema do Projeto

O projeto formulado tem como tema a utilização de uma técnica de *Machine Learning* (ML), denominada *Support Vector Regression* para utilização de dados obtidos em *software* de simulação de antenas com o intuito de prever o comprimento necessário para operação em diversas frequências de ressonância de uma antena de microfita.

1.2 Problematização

Na atualidade existe uma infinidade de métodos e técnicas de inteligência artificial, além de estudos com diferentes técnicas de *Machine Learning* propostas, como o de (3), que possuem embasamentos matemáticos diferentes e que, portanto, se adequam melhor em diferentes situações. Antenas de microfita possuem soluções analíticas, mas que não são tão confiáveis (4), já que existem diversos fatores que não são considerados nessas formulações, e para isso, foram desenvolvidos *softwares* que fazem a resolução das Equações de Maxwell para os campos, levando a valores de parâmetros, como a frequência de ressonância da antena, por exemplo, muito mais precisos e exatos aos de uma antena real se comparados a valores das equações analíticas descritas. Dito isto, deseja-se investigar se é possível predizer o valor do comprimento de uma antena de microfita para operação em diferentes frequências de ressonância, com boa precisão, utilizando como base de dados para esse modelo os próprios valores obtidos na simulação de *softwares* para antenas.

1.3 Hipóteses

As hipóteses deste trabalho são:

- É possível utilizar a *IA* para acelerar e auxiliar projetos de antenas de microsfita;
- Simulando centenas de antenas de diferentes combinações de dimensões físicas, e definindo o material de substrato e frequências trabalhados, podemos generalizar o modelo para produzir qualquer antena de microfita retangular com as características especificadas nesse texto e no intervalo de frequência de ressonância estabelecidas;
- O modelo utilizado só funcionará para a geometria especifica de antena proposta.

1.4 Objetivos

1.4.1 Objetivos Gerais

O objetivo deste projeto de pesquisa é desenvolver e treinar, com todas as etapas, um modelo de *IA* que consiga predizer o comprimento de uma antena de microfita de geometria específica utilizando informações do material e frequências de funcionamento utilizados.

1.4.2 Objetivos Específicos

- Definir e simular o projeto das antenas no *Ansys HFSS*;
- Obter dados contundentes de antenas de microfita;
- Realizar uma exploração e limpeza de dados *(Exploratory Data Analysys)*;
- Definir e treinar um modelo de *Support Vector Regression* utilizando *Scikit-Learn*;
- Avaliar o desempenho do modelo criado através de métricas estatísticas.

1.5 Justificativas

Projetos de antenas são as bases para o avanço de sistemas de telecomunicações, especialmente transmissão de dados para navegação na *internet*. Dada importância, o auxilio de uma ferramenta de *IA* é algo extremamente relevante para otimizar o tempo de criação de antenas, já que alguns projetos podem demorar horas ou até dias para serem executados (5). Caso o projeto tenha bons resultados, é possível utilizá-lo como ferramenta auxiliar para criação simples e precisa dessas antenas.

1.6 Considerações Finais

No Capítulo 1 do trabalho foram levantados pontos importantes a respeito do desenvolvimento deste trabalho para melhoria não somente das telecomunicações, como também do potencial impacto na vida das pessoas de forma geral, dada a relevância da transmissão de informação na vida atual, transmissão esta que em algum estágio passa por uma antena. A fim de melhorar as redes de telecomunicações, foi proposta uma forma

de utilizar técnicas de IA para encontrar parâmetros físicos de uma antena de microfita de forma mais fácil e confiável.

Para o próximo capítulo, um bom embasamento teórico, tanto a respeito de IA quanto a respeito de antenas será descrito, permitindo uma clara utilização de poderosas ferramentas matemáticas da SVR para chegar ao ponto alvo que é a previsão do comprimento de uma antena de microfita retangular com características específicas.

2 Fundamentação Teórica

2.1 Antena

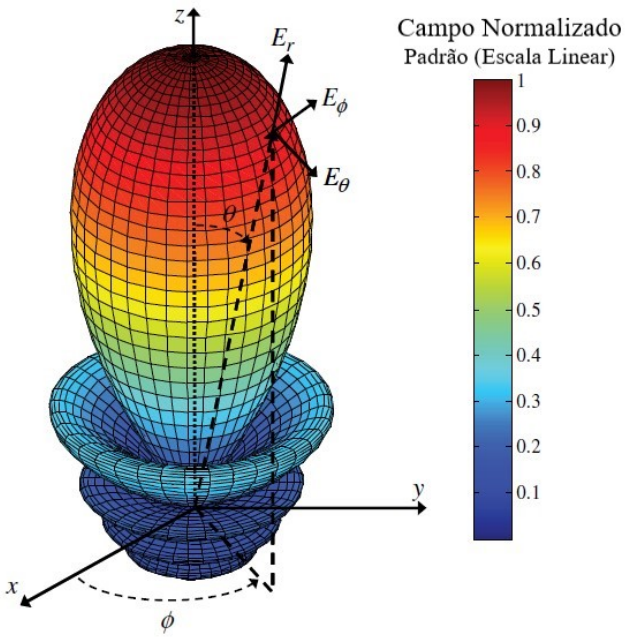
Uma antena é um dispositivo essencial em telecomunicações, já que seu objetivo é utilizar irradiação eletromagnética tanto para transportar quanto receber informação através de um meio (4), de forma a simplificar o transporte de informação e redução de custo de equipamentos. Sendo assim, a antena é um dispositivo intermediário entre a linha de transmissão, que comporta dispositivos que geram o sinal de informação e o modifica até uma forma adequada de transmissão, encaminhando-o até a antena. Depois disso, a antena então irradia esse sinal para o espaço livre, que é o canal de transporte e que introduz formas de degradação de sinal, como ruídos, atenuação, dentre outros degradantes.

Na criação de projetos de antenas, existem alguns parâmetros que são mais importantes que outros se tratando da avaliação de desempenho, sendo que alguns deles são interrelacionados e nem todos precisam ser descritos para um bom projeto (4). Dito isto, diversos desses parâmetros e formulações matemáticas que descrevem o comportamento de tais antenas se encontram a seguir.

2.1.1 Diagrama de Irradiação

O diagrama de irradiação é uma representação matemática ou gráfica que descreve o formato da irradiação eletromagnética emitida por uma antena no espaço, sendo bidimensional ou tridimensional assim como o mostrado na Figura 2.1 (4). Esse diagrama geralmente representa a magnitude de um campo como função espacial, ou seja, uma função $f(r,\theta,\phi)$ que mostra como a amplitude dos campos elétricos e magnéticos varia em função das coordenadas espaciais, mas também pode demonstrar como a potência irradiada varia no espaço, tanto em escala linear como em escala logarítmica (4).

Figura 2.1 – Diagrama de irradiação de uma antena.



Fonte: Adapatado de (4).

Esses diagramas são importantes não somente pelos motivos citados anteriormente, mas também por conter outras informações importantes, como a diretividade, polarização, fase, dentre outros parâmetros. Normalmente esses diagramas são normalizados em relação ao valor máximo, o que permite uma melhor visualização dos lóbulos secundários, e consequentemente torna mais fácil sua eliminação, já que tais lobos representam irradiação em direções não desejadas (4).

2.1.2 Intensidade de Irradiação

A intensidade de irradiação é um dos parâmetros mais importantes se tratando de qualquer tipo de irradiação, e no contexto de uma antena é definida como "a potência irradiada por unidade de ângulo sólido". O ângulo sólido é definido como aquele que, visto do centro de uma esfera de raio r , está subentendido por outra esfera com área superficial igual a de um quadrado de comprimento r (4).

Em termos de equacionamento, a intensidade de radiação (U) é dada em termos da distância (r) e da densidade de radiação (W_{rad}), assim:

$$U = r^2 W_{rad} \tag{2.1}$$

Antenas hipotéticas que conseguem irradiar igualmente em todas as direções são denominadas antenas isotrópicas, e essas antenas servem como base para a construção de outros tipos de antenas, possuindo uma certa potência irradiada (P_{rad}) , cuja intensidade de radiação é dada por:

$$U = \frac{P_{rad}}{4\pi} \tag{2.2}$$

2.1.3 Diretividade

A diretividade de uma antena é a sua capacidade de concentrar energia em determinada direção do espaço (6). Assim, quanto mais "concentrada" é a onda eletromagnética produzida pela antena, menor é o denominado ângulo de meia potência (é a abertura angular do lóbulo principal de uma antena onde a potência irradiada cai para 50é o lóbulo principal. A diretividade é dada pela Equação 2.3:

$$D = \frac{U}{U_0} = \frac{4\pi U}{P_{rad}} \tag{2.3}$$

Através de algumas manipulações algébricas e usando cálculo diferencial e integral, chega-se à equação para diretividade de antenas direcionais e baixo nível de lóbulos secundários, dado pela Expressão de Kraus na forma da Equação 2.4:

$$D \approx \frac{4\pi}{\theta_3 \phi_3} \tag{2.4}$$

sendo que θ_3 e ϕ_3 são os ângulos de meia potência dados em radianos.

2.1.4 Modo de Operação

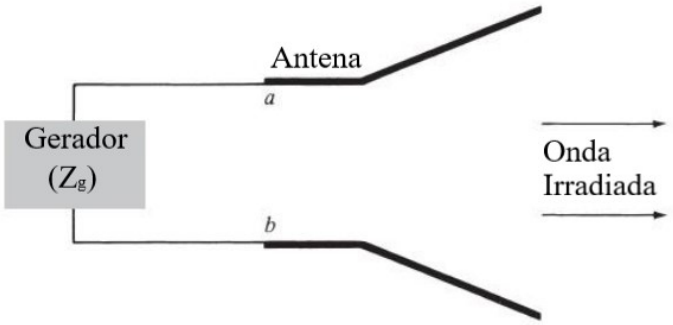
O modo de operação se aplica a antenas de ondas guiadas, grupo do qual as antenas de microfita fazem parte. Ele se refere a como as Equações de Maxwell são resolvidas dadas condições de fronteira, referindo-se, de forma física, a todas as distribuições de campo eletromagnético possíveis (7). Essas distribuições são classificadas de acordo com suas componentes vetoriais, e de acordo com a constante de fase de espaço livre β, sendo eles modos propagantes, evanescentes, modos de radiação ou também os denominados modos de vazamento (*Leaky Modes*), como mostrado a seguir:

- TEM (Transversal Eletromagnético): As componentes de campo elétrico e magnético são perpendiculares entre si e simultaneamente perpendiculares a direção de propagação da onda eletromagnética;
- TE (Transversal Elétrico): Apenas a componente de campo elétrico é perpendicular à direção de propagação da onda eletromagnética;
- TM (Transversal Magnético): A componente de campo magnético é perpendicular à direção de propagação da onda eletromagnética;
- Modos Híbridos: Modos onde existem componentes longitudinais tanto de campo elétrico quanto de campo magnético, sendo que no modo HE componentes TE são dominantes, e o modo EH, onde as componentes TM são dominantes;
- Linearmente polarizados: Suas componentes possuem amplitude arbitrária e estão defasadas entre si por $\theta = m\pi$, com m inteiro;
- Modo propagante: β real e assim possui soluções discretas;
- Modo evanescente: β puramente imaginário;
- Modo de irradiação: β real com soluções não guiadas na linha de transmissão;
- Modo de vazamento *(Leaky Modes)*: β complexo.

2.1.5 Impedância de Entrada

A impedância de entrada é um parâmetro definido como "a impedância presente nos terminais de uma antena, a proporção de tensão para corrente num par de terminais, ou ainda a proporção apropriada das componentes de campo elétrico e magnético num ponto." (4). Para antenas, a definição para pares de terminais de entrada é especialmente usual, como visto na Figura 2.2:

Figura 2.2 – Modelo de terminais para uma antena.



Fonte: adaptado de (4).

A impedância da antena pode ser definida como a tensão entre os terminais a e b sem que haja algum tipo de carga conectada nesses terminais, dada pela Equação 2.5

$$Z_A = R_A + jX_A \tag{2.5}$$

Sendo que Z_A é a impedância da antena entre os terminais $a - b$ (ohms), R_A é a resistência entre os terminais $a - b$ (ohms) e X_A é a reatância entre os terminais $a - b$ (ohms). Quando a parte imaginária é nula diz-se que a antena está na condição de ressonância, desse modo a impedância de entrada é puramente real. A parte real na Equação 2.5 é dividida em duas componentes, uma que se refere a resistência de irradiação da antena (R_r)e outra se referindo a resistência de perda da antena (R_L), ocasionada por fatores como efeito Joule, descasamento de impedância com a carga, dentre outros. Essas resistências são descritas pela Equação 2.6:

$$R_A = R_r + R_L \tag{2.6}$$

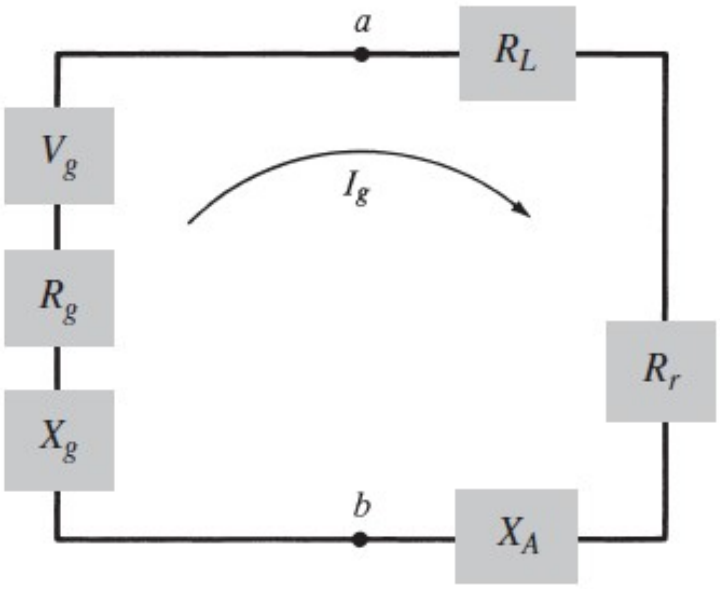
Quando uma antena está operando, principalmente em transmissão, geralmente ela está conectada a um gerador, cuja impedância interna também deve ser considerada, dada pela Equação 2.7.

$$Z_g = R_g + X_g \tag{2.7}$$

Sendo que Z_g é a impedância do gerador (ohms), R_g é a resistência da impedância do gerador (ohms) e X_g é a reatância da impedância do gerador (ohms).

Em modo de transmissão, a antena e o gerador podem ser simplificados a um circuito equivalente de Thevenin, tal como visto na Figura 2.3:

Figura 2.3 – Equivalente de Thevenin para o circuito da antena e do gerador.



Fonte: (4).

Sendo assim, é possível encontrar a potência entregue a antena para irradiação P_r utilizando conceitos de simplificação de circuitos elétricos juntamente com a 1ª Lei de Ohm e a Lei da Potência, dada pela Equação 2.8:

$$P_R = \frac{1}{2}|I_g|^2 R_r = \frac{|V_g|}{2}\left[\frac{R_r}{(R_r + R_L + R_g)^2 + (X_A + X_g)^2}\right](W) \tag{2.8}$$

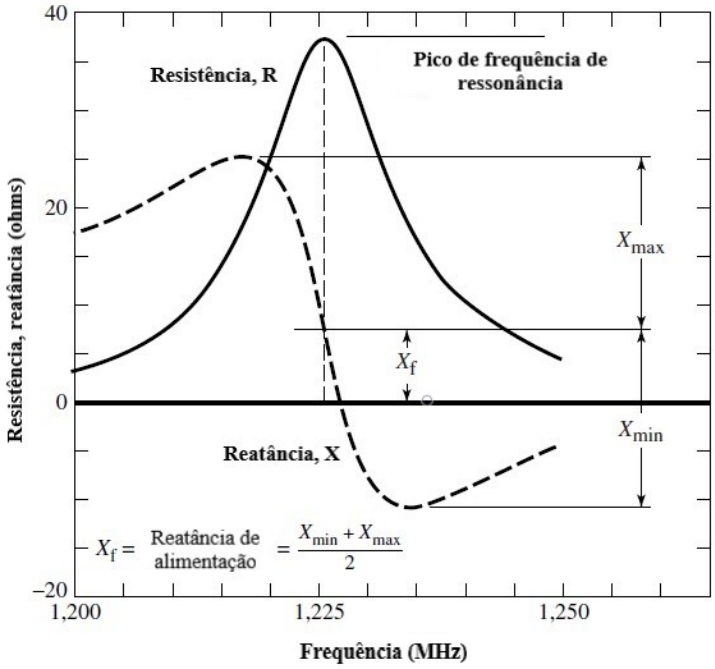
De forma que I_g é a corrente fornecida pelo gerador e V_g é a tensão do gerador. É importante ressaltar que a Equação 2.8 é válida para antenas mais simples e em situações em que não há perdas no dielétrico ou no plano terra.

Como visto na Equação 2.5 a impedância de entrada possui uma parte real e uma parte imaginária. Um exemplo disso é apresentado na Figura 2.4 para antenas de microfita via sonda coaxial. Nesse caso, a impedância de entrada é aproximada e leva em consideração diversos fatores. Isso acontece por causa da sobreposição de campos de alimentação e da antena em si, causados pela posição, a reatância e o comprimento da sonda de alimentação, dentre outros fatores (4). Observa-se que a estrutura de alimentação gera um efeito indutivo, deslocando o ponto de ressonância a x_f ohms como apresentado na Figura 2.4. Esse valor é quantificado na Equação 2.9, em que d é o diâmetro da sonda

de alimentação, h é a espessura do substrato, η é a impedância intrínseca e k é o número de onda:

$$x_f \approx -\frac{\eta k h}{2\pi}\left[ln\left(\frac{kd}{4}\right) + 0.577\right] \tag{2.9}$$

Figura 2.4 – Curva típica da resistência e da reatância numa antena de microfita retangular.

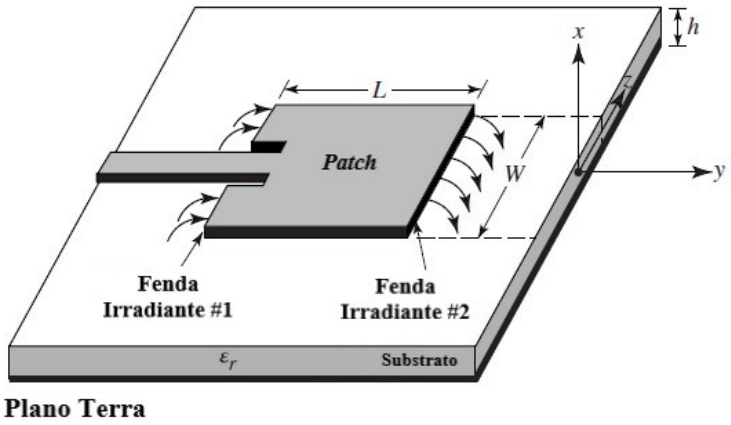


Fonte: adapatado de (4).

2.2 Antenas de Microfita

Antenas de microfita (*Microstrip Patch Antennas*) surgiram da necessidade de se criar antenas que têm uma boa operabilidade mesmo quando se reduz suas dimensões físicas, podendo ser utilizadas em aplicações de antenas compactas para celulares, aeronaves, e diversos sistemas de comunicação nos quais tamanho e custo são fatores cruciais (6). Esse nome se deve ao fato de a antena básica ser uma área retangular conectada a um ponto de alimentação estreito e montada sobre um substrato, que por fim é montado sobre um plano terra, como mostrado na Figura 2.5:

Figura 2.5 – Forma geral de uma antena de microfita.

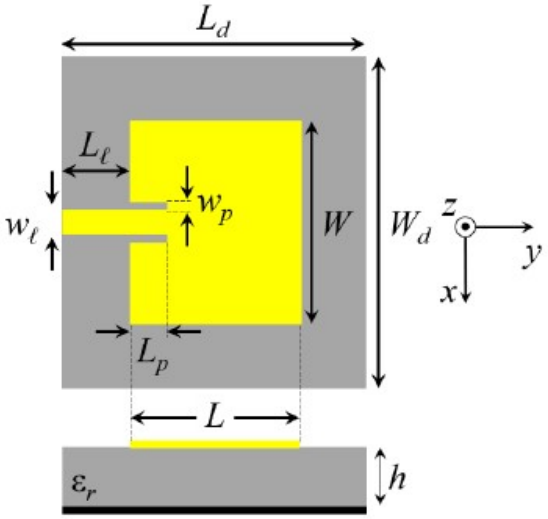


Fonte: (4).

Tais antenas possuem diversas simetrias e formatos diferentes, como retangular, circular, quadrada, elíptica, e diversos outros, sendo aquele utilizado dependente da aplicação, parâmetros físicos, dificuldade de análise e forma de alimentação (8).

Antenas de forma retangular como a da Figura 2.6 são mais simples de serem modeladas, e por isso são os tipos base para esse trabalho, além de geralmente operarem próximas à frequência de ressonância. Seus parâmetros são: largura do elemento ressonador (W), comprimento do elemento ressonador (L), comprimento e largura dos cortes casadores de impedância no elemento irradiador (L_p e w_p), comprimento e largura do dielétrico (L_d e W_d), a espessura do substrato (h) e o comprimento e largura da linha de alimentação (L_l e w_l) (9).

Figura 2.6 – Antena de microfita retangular.



Fonte: (9).

Para o equacionamento de antenas de microfita, é necessária a resolução das Equações de Maxwell. Entretanto, para o caso destes circuitos, não há uma solução analítica precisa de tais equações, o que leva à utilização de 3 formas diferentes para enfrentar esse problema. A primeira dessas soluções é a utilização de métodos numéricos, feitas através de *softwares* específicos que efetuam essas operações com resultado aproximado. A segunda ferramenta é o denominado método da cavidade ressonante, que interpreta a antena como uma cavidade ressonadora delimitada pelas dimensões dessa antena. Por último, o método da linha de transmissão, que modela a antena como uma linha de transmissão que irradia as ondas eletromagnéticas pelas suas extremidades.

Outro fator importante de ser citado, é que a maioria das dimensões da antena citadas, como largura do dielétrico e da linha de alimentação são resumidos a proporções de dimensões mais usuais, sendo assim uma forma de simplificar o problema, reduzindo a quantidade de variáveis do problema.

2.2.1 Comprimento do elemento ressonador

O comprimento do elemento ressonador é, na verdade, um valor de comprimento inicial corrigido por um acréscimo de comprimento causado pelos efeitos de "franja" dos campos que atuam sobre a antena (ΔL), dado pela Equação 2.10:

$$L = \frac{1}{2f_r\sqrt{\epsilon_{ef}}\sqrt{\mu_0\epsilon_0}} - 2\Delta L \tag{2.10}$$

O valor de correção pode ser dado como um quociente $\frac{\Delta L}{h}$ ou também como simplesmente um valor ΔL como na Equação 2.11

$$\Delta L = 0.412h\left[\frac{(\epsilon_{ef}+0.3)\left(\frac{W}{h}+0.264\right)}{(\epsilon_{ef}-0.258)\left(\frac{W}{h}+0.8\right)}\right] \tag{2.11}$$

Já o valor de ϵ_{ef}, que é a constante dielétrica efetiva, é calculado pela Equação 2.12

$$\epsilon_{ef} = \frac{\epsilon_r+1}{2} + \left[\frac{\epsilon_r-1}{2}\left(1+12\frac{h}{W}\right)^{-\frac{1}{2}}\right] \text{ sendo que } \frac{W}{h} > 1 \tag{2.12}$$

De forma que ϵ_r é a constante dielétrica relativa.

2.2.2 Largura do elemento ressonador

A largura do elemento é obtida assumindo-se o modo fundamental de operação TM_{10}, de forma que essa dimensão é normalizada, é dada pela Equação 2.13:

$$W = \frac{1}{2f_r\sqrt{\mu_0\epsilon_0}}\sqrt{\frac{2}{\epsilon_{ef}+1}} = \frac{c}{2f_r}\sqrt{\frac{2}{\epsilon_{ef}+1}} \tag{2.13}$$

2.2.3 Condutância

A condutância de uma antena é definida como a sua capacidade em conduzir corrente elétrica, ou seja, o inverso de sua impedância. A condutância é expressa pela Equação 2.14.

$$G_1 = \frac{2P_{rad}}{|V_0|^2} = \frac{I_1}{120\pi^2} \tag{2.14}$$

O valor de I_1 é dado pela Equação 2.15.

$$I_1 = \int_0^{\pi}\left[\left[\frac{sen\left(\frac{\kappa_0 W}{2}cos\theta\right)}{cos\theta}\right]^2 sen^3\theta d\theta\right] \tag{2.15}$$

2.2.4 Potência irradiada

Numa antena de microfita, a potência radiada denota uma determinada potência que atravessa uma superfície fechada (4) dado pela Equação 2.16.

$$P_{rad} = \frac{|V_0|^2}{2\pi\eta_0}\int_0^{\pi}\left[\left[\frac{sen\left(\frac{\kappa_0 W}{2}cos\theta\right)}{cos\theta}\right]^2 sen^3\theta d\theta\right] \tag{2.16}$$

É importante salientar que o padrão de radiação desse tipo de antena é bastante largo e com direção máxima no plano normal da antena, o que impacta diretamente na computação numérica de dito padrão (8).

2.2.5 Frequência de Ressonância

Um dos tópicos mais importantes quando se fala de antenas em geral, é a frequência de ressonância, pois ela indica onde a antena opera com máxima eficiência (4).

Para encontrar um valor aproximado para a frequência de ressonância, umas das melhores formas é assumir que os campos no substrato elétrico é modelado como se ele estivesse numa cavidade (com condutores elétricos abaixo e acima) e também por "paredes" magnéticas.

A equação geral dada para frequências de ressonância em modos TM gerais com m, n e $p \in \mathbb{Z}_+$ é dado pela Equação 2.17:

$$f_{(r)\,mnp} = \frac{1}{2\pi\sqrt{\mu\epsilon}}\sqrt{\left(\frac{m\pi}{h}\right)^2 + \left(\frac{n\pi}{L}\right)^2 + \left(\frac{p\pi}{W}\right)^2} \tag{2.17}$$

Como o projeto da antena é feita para o modo TM_{10}, ou seja, $m = 0$, $n = 1$ e $p = 0$, a Equação 2.17 se simplifica a equação Equação 2.18

$$f_{(r)\,mnp} = \frac{v_0}{2L\sqrt{\epsilon_r}} \tag{2.18}$$

Onde v_0 é a velocidade da luz no espaço livre.
L é o comprimento da antena.
ϵ_r é a permissividade relativa do material dielétrico.

2.2.6 Diagrama de irradiação para antenas de microfita

O padrão de radiação em antena de microfita é obtido ao se representar o campo elétrico das franjas na superfície normal à antena e levando-se em conta o efeito de imagem da corrente magnética no plano terra quando t $\rightarrow$ 0, ou seja, $M_s = 2E_a \times \vec{n}$, sendo que E_a é o campo elétrico nas franjas e $\vec{n}$ é o vetor normal à antena(8). Os campos elétricos azimutais e de elevação são dados pelas Equação 2.19 e Equação 2.20, respectivamente.

$$E_\theta = E_0 cos\,(\phi)\, f\,(\theta, \phi) \tag{2.19}$$

$$E_\phi = -E_0 cos\,(\theta)\, sen\,(\phi)\, f\,(\theta, \phi) \tag{2.20}$$

Sendo que, o valor de $f\,(\theta,\phi)$ é dado pela Equação 2.21

$$f\,(\theta,\phi) = \left[\frac{sen\left(\frac{\beta W}{2} sen\,(\theta)\, sen\,(\phi)\right)}{\left(\frac{\beta W}{2} sen\,(\theta)\, sen\,(\phi)\right)}\right] \tag{2.21}$$

Onde β é a constante de fase de espaço livre.

Entender os parametros citados nessa subseção é imprenscidível para um projeto correto e funcional de uma antena de microfita.

2.3 *Machine Learning*

Machine Learning ou Aprendizado de Máquina é uma aplicação de inteligência artificial na qual um *software* consegue aprender diversos aspectos sobre determinado assunto, sem que instruções explícitas sejam dadas ao programa, através de maior exposição e com uma melhor otimização de tempo (10).Muitas vezes o ML é confundido com o *Deep Learning (DL)*, mas na verdade, o ML é uma técnica de IA, e o DL é uma técnica de ML. Um exemplo de algoritmo pertencente ao DL são as *Deep Neural Networks (DNN)*, que são, por sua vez, algoritmos que buscam imitar o funcionamento de neurônios. Eles resolvem tarefas complexas através da interconexão de diversos "nós", tal que um estímulo na entrada desses nós desencadeiam uma ação por determinado caminho nessa rede (11).

Existem tipos diferentes de ML que funcionam de formas diferentes, necessitando ou não de supervisão e rotulação de dados, existindo diversas técnicas diferentes para cada tipo de ML, baseando-se em diferentes modelos estatísticos, classificação e treinamento de dados, sendo possível categorizá-las por:

- Aprendizado supervisionado: Nesse tipo de ML, temos um conjunto de entradas cedidas a um modelo de treinamento, rotulando essas entradas e algumas saídas corretas também cedidas ao modelo, seja para classificação ou predição de dados na saída (12).Como exemplos desse tipo de técnica podemos citar classificadores lineares, *K-Nearest Neighbors (KNN)*, *Decision Tree*, *Support Vector Machines (SVM)*, dentre outras;

- Aprendizado não-supervisionado: Com essa técnica de ML, não há uma rotulagem dos dados, sendo assim o modelo deve descobrir padrões e semelhanças entre os dados cedidos, comparar com a entrada, e encontrar a melhor saída possível (13). Nessa categoria se incluem técnicas como *K-Means Clustering (K-Means)* e *Gaussian Mixture Models (GMM)*;

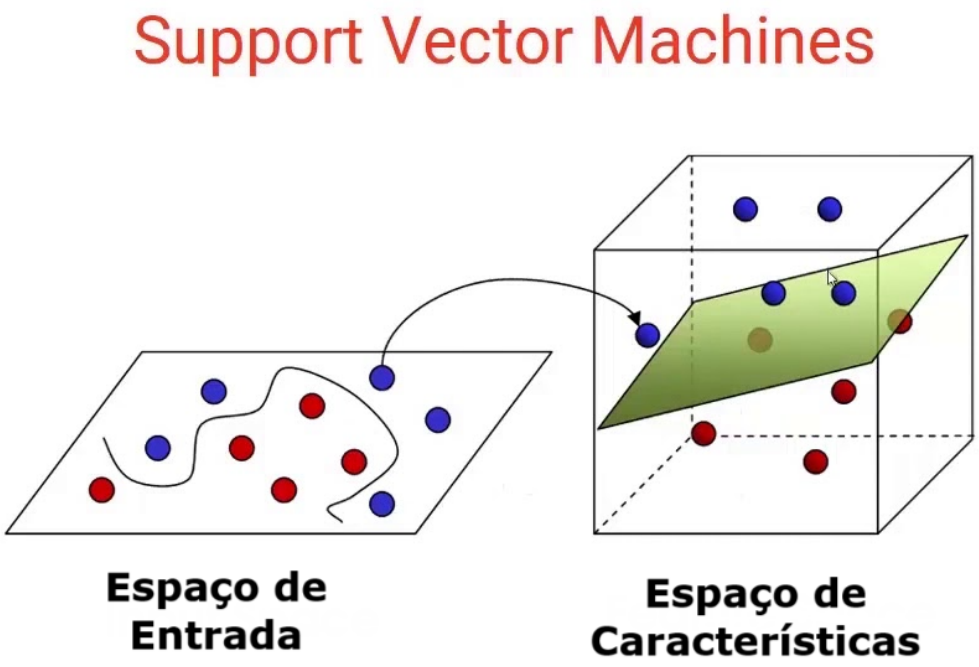
- Aprendizado semi-supervisionado: Esse tipo de aprendizado é uma técnica de ML na qual alguns poucos dados são rotulados e o restante não, aprimorando a capacidade

dos modelos desse tipo em descobrir o comportamento dos dados pela distribuição estatística. Um ponto desfavorável, no entanto, é que se torna necessário fazer suposições sobre essa distribuição (14).

2.3.1 *Support Vector Machines*

Support Vector Machine é uma técnica de ML supervisionada, baseada em aprendizado estatístico desenvolvida por Vapnik (15) estabelecendo diversos princípios para que uma boa generalização do modelo para os dados possa ser obtida. A técnica utiliza diversos tipos de hiperplanos, que são uma espécie de "fronteira" para classificação de dados, ou seja, se os dados analisados pertencem a um grupo ou a outro, sejam essas fronteiras lineares ou não, com N dimensões, como podemos ver na Figura 2.7:

Figura 2.7 – Exemplos de hiperplanos



Fonte: (16).

O problema de aprendizado supervisionado é dado matematicamente como um conjunto de dados de entrada do tipo $(x_1,y_1), (x_2,y_2), ...,(x_n,y_n)$ no espaço $\mathbb{R}^n \times \mathbb{R}$ com uma distribuição de probabilidade desconhecida $P(x,y)$ e uma função de perda $V(y,f(x))$ que funciona como uma forma quantitativa de medir o erro de quando, para um dado valor x o valor $f(x)$ é previsto ao invés do valor y tornando-se necessário encontrar uma função f que minimize esse erro (17).

Dada essa formulação e o fato de que a distribuição de probabilidade $P(x,y)$ é desconhecida, o Princípio de Minimização de Risco Empírico é utilizado para descobrir quais são as possíveis funções f que satisfazem a hipótese de minimização do erro para um espaço de hipótese dado por H, como mostra (17), dado por:

$$H = \frac{1}{l}\sum_{i=1}^{l} V\left(y_i, f\left(x_i\right)\right) \tag{2.22}$$

É possível melhorar o erro empírico considerando estruturas de espaços de erros considerando i espaços H diferentes, cada um com complexidade maior conforme o espaço i aumenta. Dito isto, (18) em seu primeiro teorema para aprendizado estatístico diz que "Se V é a V_c-ésima dimensão de um espaço de hipóteses H, então para uma probabilidade $1-\eta$, o valor mínimo de erro esperado que pode ser alcançado com funções do espaço H, L_{err_min} por exemplo, e o mínimo erro empírico emp L_{emp} devem satisfazer a inequação:"

$$L_{emp} - 4\sqrt{2}\sqrt{\left[\frac{V\left(1+log\left(\frac{2l}{V}\right)\right)-log\left(\frac{\eta}{4}\right)}{l}\right]} \leq L_{err\_min} \leq L_{emp} + 4\sqrt{2}\sqrt{\left[\frac{V\left(1+log\left(\frac{2l}{V}\right)\right)-log\left(\frac{\eta}{4}\right)}{l}\right]} \tag{2.23}$$

Sendo que a Equação 2.23 é independente da distribuição de probabilidade $P(x,y)$.

Para que um algoritmo de SVM possa realizar essa ideia de minimizar o erro utilizando a formulação anterior, tanto o espaço de hipóteses H quanto a função de perda $V(y,f(x))$ devem ser estabelecidas. Cabe então ao algoritmo de SVM estabelecer um hiperplano adequado para o vetor de dados de entrada $\boldsymbol{x}$ que resolva esse problema. Um exemplo seria uma reta, de maneira que o espaço de hipóteses seria qualquer subespaço que contenha uma função $f(x)$ da forma da Equação 2.24:

$$f(x) = w.\boldsymbol{x} + b \tag{2.24}$$

Geralmente, a SVM utiliza um hiperplano diferente do hiperplano que contém o vetor de dados $\boldsymbol{x}$, definido por um *kernel* K, onde esse *kernel* pode ser pensado como um hiperplano que define um produto escalar com o vetor $\boldsymbol{x}$ (19), definindo então o espaço de hipóteses como um conjunto de hiperplanos criados através de K. Computacionalmente, é feita uma relação de compromisso entre a complexidade do plano de hipóteses e o erro empírico, relação essa que pode ser obtida efetuando a minimização de um problema de

classificação e um problema de regressão, dados respectivamente pela Equação 2.25 e Equação 2.26:

$$\underset{f}{min}\|f\|_k^2 + C\sum_{i=1}^{l}|1 - y_i f(x_i)|_+ \tag{2.25}$$

$$\underset{f}{min}\|f\|_k^2 + C\sum_{i=1}^{l}|y_i - f(x_i)|_\epsilon \tag{2.26}$$

Em que C é um parâmetro que controla a relação de compromisso mencionada.

Os problemas de regressão e de classificação mencionados acima podem ser reescritos e solucionados como um único problema de programação quadrática restrita (PQ), que para o caso do problema de classificação, a forma de resolução sugerida por (17) é a utilização do Método dos Pontos Interiores, obtendo o problema de formulação dupla da Equação 2.27:

$$\underset{\alpha_i}{min}\sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i=1}^{1}\sum_{j=1}^{1}\alpha_i y_j \alpha_j y_i K(x_i, x_j) \tag{2.27}$$

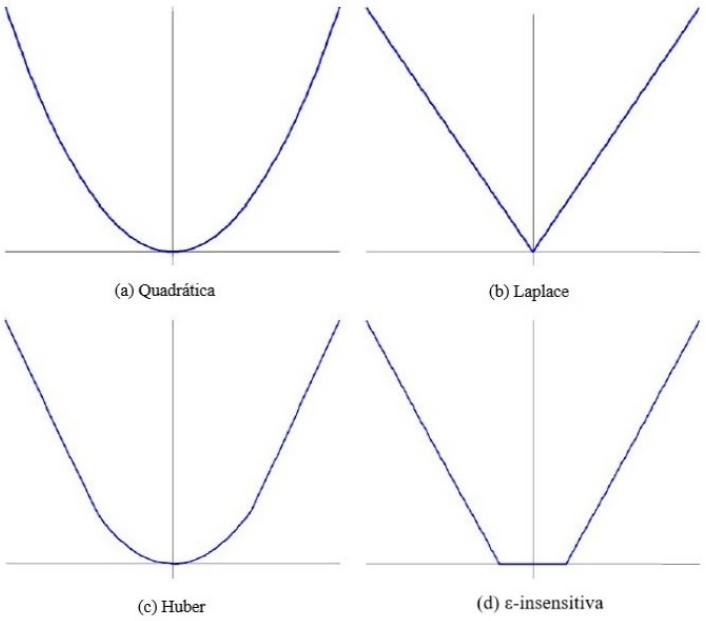
Em que α_i deve ficar confinada no intervalo $0 \leq \alpha_i \leq C$ para todo valor de i e também temos a consdição da Equação 2.28:

$$\sum_{j=1}^{l}\alpha_i y_i = 0 \tag{2.28}$$

2.3.2 *Support Vector Regression*

Utilizando a base matemática da SVM é possível criar uma estrutura semelhante a esta, modificando a função de perda (20) a fim de incluir a distância medida, sendo assim possível executar tarefas de regressão de dados. Como exemplos de algumas das funções de perda utilizadas, pode-se citar funções quadráticas, funções de Huber, funções ϵ-insensitivas, funções Laplacianas, como visto na Figura 2.8

Figura 2.8 – Exemplos de funções de perda (a)quadrática (b)Laplaciana (c)de Huber e (d)ϵ-insensitiva.



Fonte: (20).

A utilização das funções de perda depende de parâmetros estatísticos acerca dos dados, como por exemplo a sensibilidade a valores atípicos dentro desses dados, também denominados de *outliers*, se a base de distribuição probabilística é conhecida ou não, e assim em diante. Entretanto, as três primeiras funções da Figura 2.8 não criam algo denominado esparsidade nos vetores de suporte. A fim de contornar esse problema, Vapnik criou a função ϵ-insensitivas na Figura 2.8 (d).

A regressão se trata de aproximar um conjunto de dados discretos pela melhor função que minimiza a soma do quadrado dos resíduos (21), de cada ponto até essa função. Além disso, possibilita uma melhor análise estatística acerca dos dados como um todo, como por exemplo a correlação e predizer, para um dado valor de entrada $\boldsymbol{x}$ qual a melhor saída correspondente y. Existem, no entanto, regressões lineares e não lineares, que consistem na aproximação por uma função linear ou não linear, de acordo com a complexidade dos dados.

Numa regressão linear, é necessário aproximar um dado conjunto de dados polinomiais $D = \{(x^1,y^1),(x^2,y^2),...,\left(x^l,y^l\right)\}$ com $\boldsymbol{x} \in \mathbb{R}^n$, $y \in \mathbb{R}$ por uma função linear da forma da Equação 2.29:

$$f(x) = \langle w,\boldsymbol{x}\rangle + b \tag{2.29}$$

em que $\langle\rangle$ representa o espaço do produto interno entre w e $\boldsymbol{x}$.

A função de regressão que melhor se adequa como solução para esse problema envolve encontrar o mínimo da Equação 2.30.

$$\Phi(w,\xi) = \frac{1}{2}\|w\|^2 + C\sum_i \left(\xi_i^- + \xi_i^+\right) \tag{2.30}$$

em que ξ_i^- e ξ_i^+ são variáveis que restringem o problema em limites inferiores e superiores, respectivamente, transformando uma inequação numa equação e C é um valor prédeterminado.

A melhor função que se adequa ao modelo de SVR, como já dito, é a função ϵ-insensitiva , que é uma aproximação da função de Huber vista na Figura 2.8 (c) que introduz uma maior esparsidade aos vetores suporte, e que por definição é dada pela Equação 2.31:

$$L_\epsilon(y) = \begin{cases} 0, & \text{para } |f(x) - y| < \epsilon \\ |f(x) - y| - \epsilon, & \text{caso contrário} \end{cases} \tag{2.31}$$

A função ϵ-insensitiva então tenta realizar uma regressão na maior dimensão do espaço, ao mesmo tempo em que tenta minimizar a complexidade do modelo, minimizando o parâmetro $||w||^2$ conforme descrito anteriormente na Equação 2.31.

A solução da Equação 2.31 é dado pela Equação 2.32:

$$\underset{\alpha,\alpha^*}{max}\, W(\alpha,\alpha^*) = \underset{\alpha,\alpha^*}{max} -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_i - \alpha_i^*)\langle x_i, x_j\rangle + \sum_{i=1}^{l}\alpha_i(y_i - \epsilon) - \alpha_i^*(y_i + \epsilon) \tag{2.32}$$

Em que tal solução possui as restrições impostas pela Equação 2.33 e pela Equação 2.34:

$$0 \leq \alpha_i, \alpha^* \leq C \qquad\qquad i = 1,2,...,l \tag{2.33}$$

$$\sum_{i=1}^{l}(\alpha_i - \alpha^*) = 0 \tag{2.34}$$

Resolvendo a Equação 2.32 com as restrições da Equação 2.33 e da Equação 2.34, pode-se determinar α e α^*, que são os multiplicadores de Lagrange dessa equação. Com esses valores pode-se encontrar os máximos e mínimos desse problema de otimização, e, portanto, a regressão que melhor se adequa ao problema da Equação 2.29 que é dada pelos valores $\bar{w}$ e $\bar{b}$, dados pela Equação 2.35 e Equação 2.36, respectivamente:

$$\bar{w} = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*)\, x_i \tag{2.35}$$

$$\bar{b} = -\frac{1}{2} \langle \bar{w}, (x_r + x_s) \rangle \tag{2.36}$$

As condições de Karush-Kuhn-Tucker (KKT), que nada mais são que testes de primeira derivada, as quais determinam se uma solução de programação não linear é ótima ou não (22) são dadas pela Equação 2.37:

$$\bar{\alpha_i}\bar{\alpha_i}^* = 0 \qquad\qquad i = 1,2,...,l \tag{2.37}$$

Dessa forma, os vetores de suporte devem ser pontos nos quais um dos multiplicadores de Lagrange é maior que zero. Tornando $\epsilon = 0$ a função de perda L_1 pode ser encontrada, o que reduz o problema de otimização a Equação 2.38:

$$\underset{\beta}{min} \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \beta_i \beta_j \langle x_i, x_j \rangle - \sum_{i=1}^{l} \beta_i y_i \tag{2.38}$$

De tal forma que a Equação 2.38 é limitada pelas condições impostas na Equação 2.39 e na Equação 2.40:

$$-C \leq \beta_i \leq C \qquad\qquad i = 1,2,...,l \tag{2.39}$$

$$\sum_{i=1}^{l} \beta_i = 0 \tag{2.40}$$

Com isso, finalmente é possível determinar a função de regressão adequada ao problema da Equação 2.29 através dos valores de $\bar{w}$, dado pela Equação 2.41, e $\bar{b}$ pela Equação 2.42

$$\bar{w} = \sum_{i=1}^{l} \beta_i = 0 \tag{2.41}$$

$$\bar{b} = -\frac{1}{2} \langle \bar{w}, x_r + x_s () \rangle \tag{2.42}$$

2.4 Considerações Finais

Neste capítulo foram abordadas tanto a modelagem física quanto a matemática por trás de antenas de microfita de forma geral, desenvolvendo as características e parâmetros que influenciam em tal equacionamento, além de abordar o aspecto de simulações com antenas. Na parte referente ao modelo de ML a ser utilizado, é discorrido sobre a teoria matemática central na qual a SVM e a SVR se baseiam, e a forma como essas técnicas são incorporadas em algoritmos, de forma que os computadores possam entendê-las e aplicá-las.

3 Materiais e Métodos

3.1 Metodologia de Desenvolvimento

A metodologia do trabalho envolve o projeto de uma antena em um simulador, estabelecendo os materiais que são utilizados para confeccionar a antena, e consequentemente sua permissividade específica ϵ_r e também a definição da frequência de operação f_r da antena, prevendo o comprimento L e a largura W da antena. O trabalho adota os seguintes passos:

1. Instalação do software simulador eletromagnético *Ansys Electronics Desktop* no computador;

2. Definição de uma antena padrão inicial para servir como base;

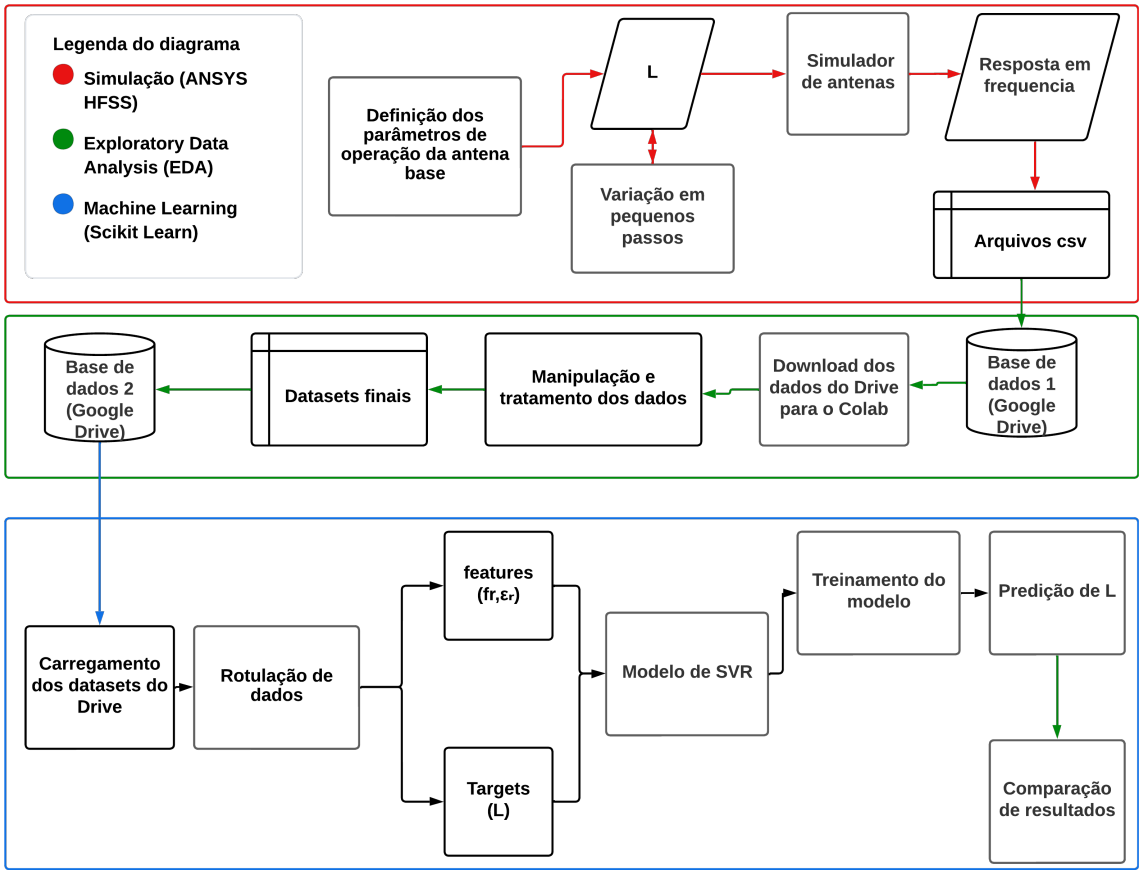
3. Tendo como base os parâmetros de entrada, sendo eles a frequência de ressonância da antena (f_r) e a constante dielétrica (ϵ_r) , além do parâmetro de saída (L) definidos, obter uma base de dados grande o suficiente para que haja uma boa rotulação dos dados para alimentar o modelo. Varia-se o comprimento L da antena em pequenos passos, obtendo-se uma resposta em frequência para combinação de L, ϵ_r e W como saída, e então adiciona-se tanto os valores de entrada como saída a uma base de dados;

4. Utilizar os dados obtidos no passo 3 para alimentar um modelo de SVR, utilizando a linguagen Python, com a biblioteca *Scikit-Learn*;

5. Obter uma amostra aleatória utilizando o preditor com o modelo construído e simular novamente no *Ansys* para comparação de resultados.

O funcionamento completo pode ser visto na Figura 3.1.

Figura 3.1 – Diagrama de metodologia.



Fonte: o autor.

3.2 Recursos Necessários

São utilizados para a conclusão deste trabalho diversos recursos, principalmente no que tange a diferentes plataformas computacionais para etapas específicas, sendo elas:

- Computador;
- *Software ANSYS HFSS* instalado no computador;
- Plataforma para programação em Python (Google Colab, Jupyter Notebook, etc.);
- Instalação de biblioteca para manipulação de dados tabulares e computação numérica (Pandas e Numpy);
- Instalação de biblioteca para *Machine Learning*, neste caso *Scikit Learn*;

3.3 Considerações Finais

Neste capítulo foram descritos todos os materiais necessários para conclusão do trabalho, bem como uma metodologia detalhada a respeito de como o projeto foi desenvolvido.

4 Desenvolvimento

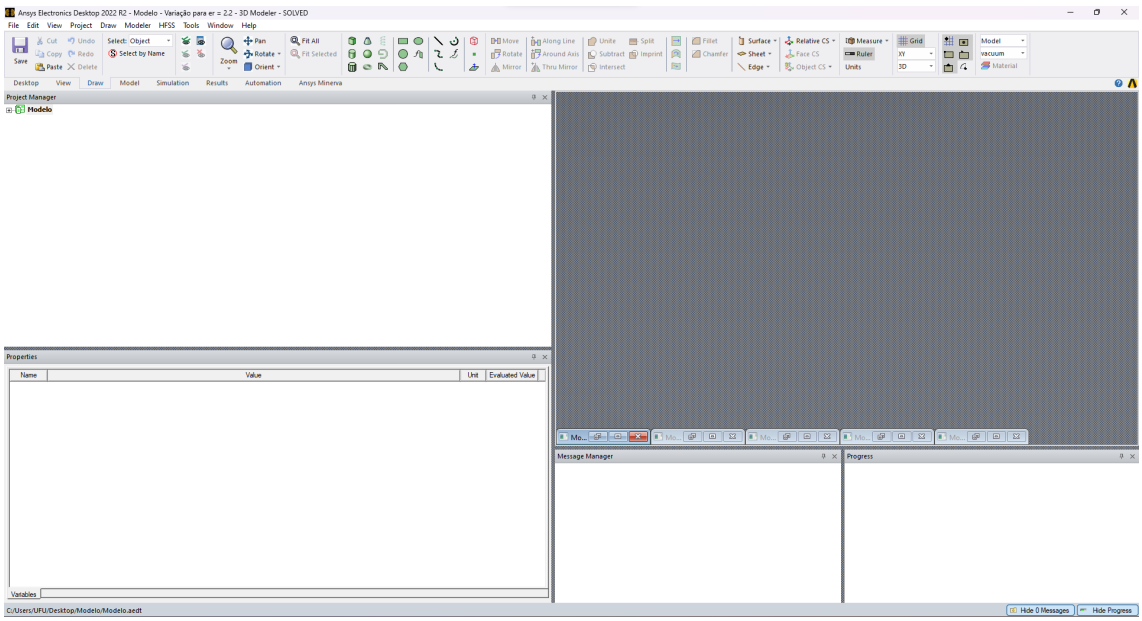
Neste capítulo são apresentados todos os passos para desenvolvimento do trabalho, incluindo definição dos parâmetros e formas das antenas, simulação completa do comportamento da antena de acordo com a resolução das Equações de Maxwell, Exploratory Data Analysis e treinamento, avaliação e outras considerações sobre o modelo de *SVR* implementado.

4.1 Processo de Criação e Simulação de Antenas

Um aspecto importante desse projeto é que todo balanceamento e equacionamento, a fim de se encontrar a melhorer estrutura da antena, foi baseado em (9), fazendo adequações a para que seja possível a variação das dimensões da estrutura.

O passo inicial no desenvolvimento do trabalho envolveu a criação de um modelo adequado para simulação de antenas de microfita no *software ANSYS Electronics Desktop 2022* através da utilização do módulo *HFSS*. A tela inicial do aplicativo é semelhante a tal qual é mostrada na Figura 4.1.

Figura 4.1 – Tela inicial do *ANSYS 2022*.

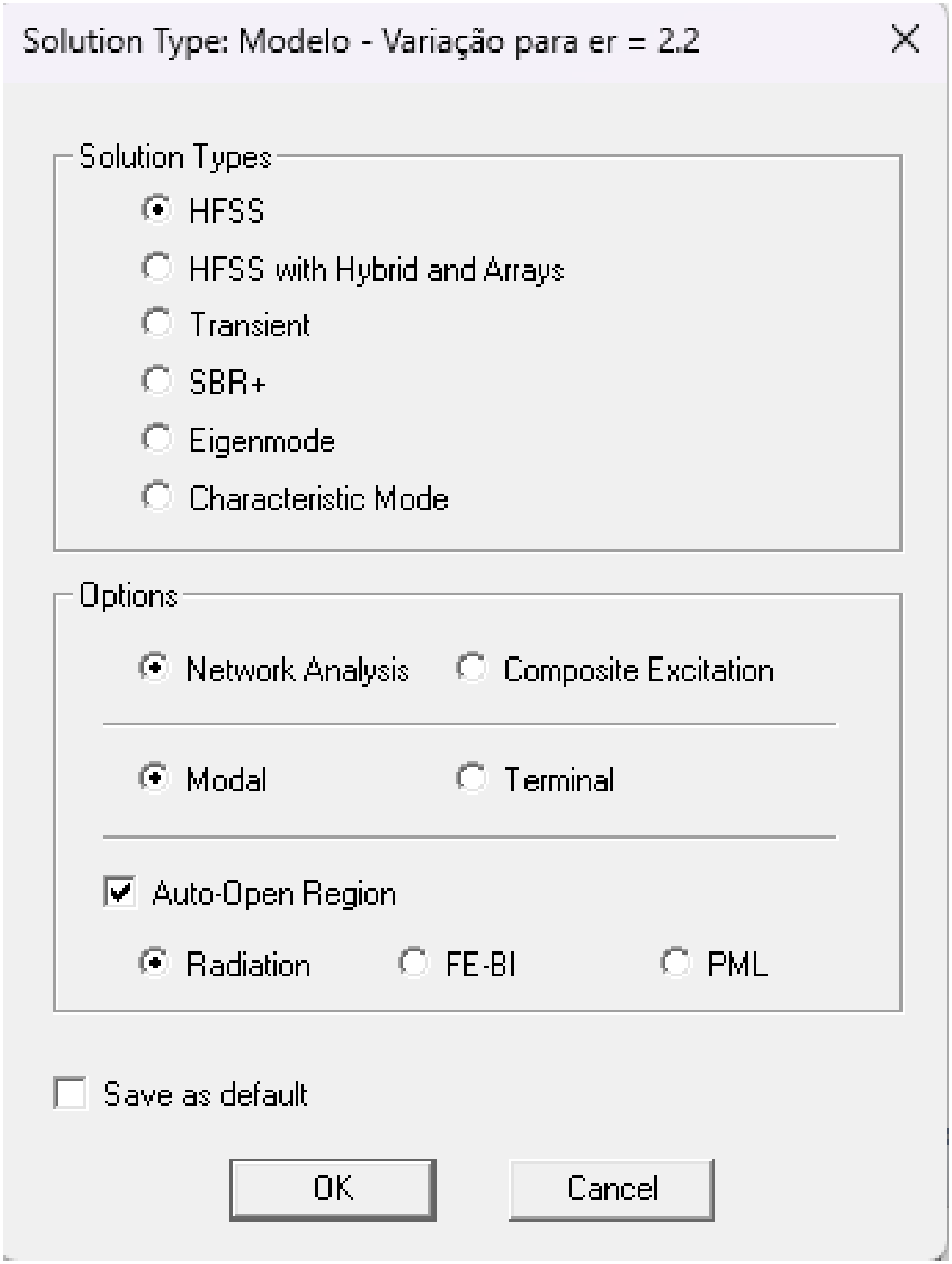


Fonte: o autor.

Para tal tarefa é necessário, primeiramente, a criação de um novo projeto e então definir características básicas para a simulação. A primeira dessas características é a

definição do tipo de análise a ser executada. Para esse projeto, foi utilizada a *Solution Type* do tipo *HFSS* com *Network Analysis* do tipo modal, que se resume a obter as soluções do problema pela utilização dos modos de propagação, como mostrado na Figura 4.2.

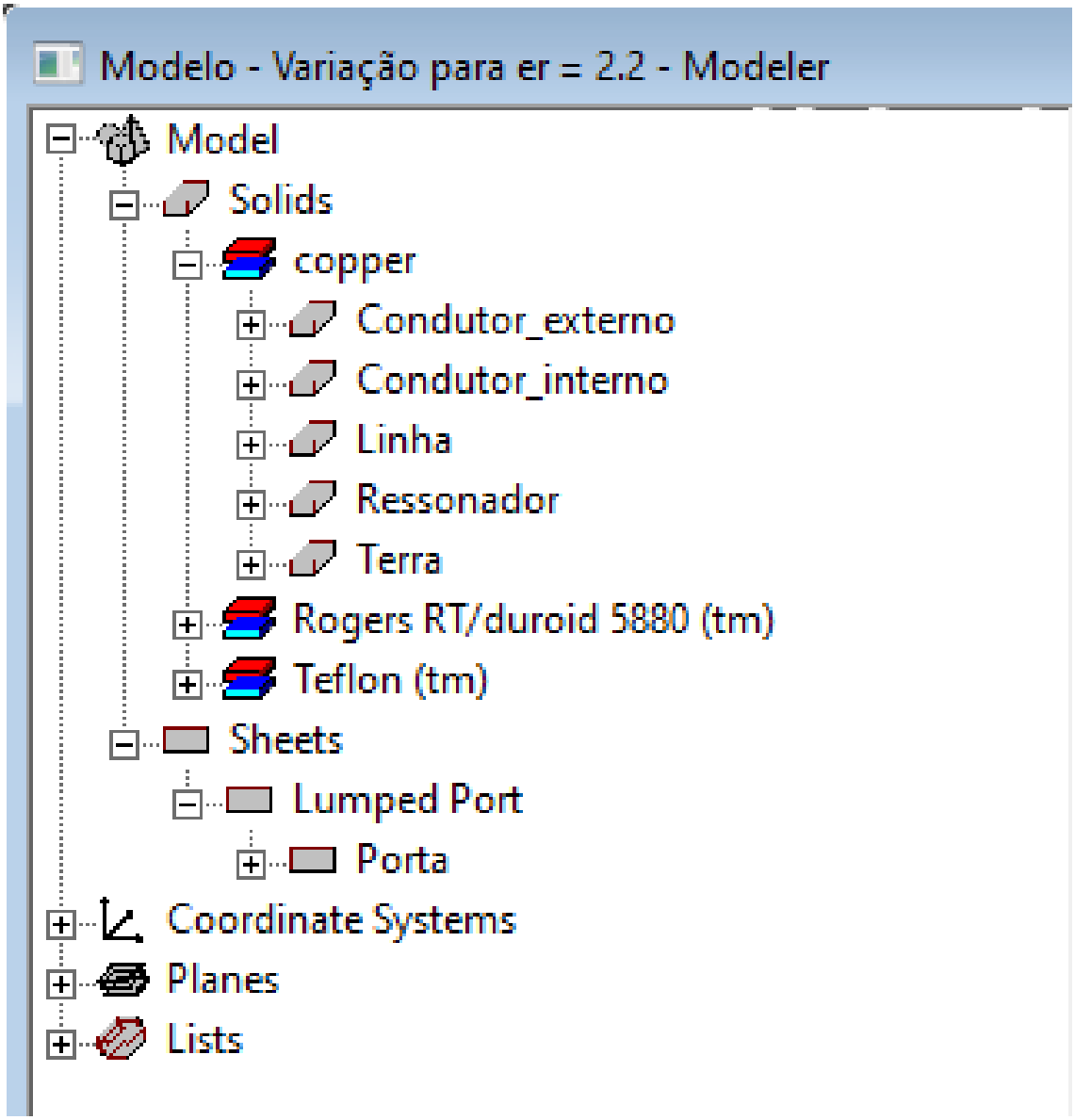
Figura 4.2 – Configurações de tipos de solução do *Ansys*.



Fonte: o autor.

O próximo passo envolve a criação de uma geometria para o dispositivo, sendo possível a adição, substituição e extração de formas geométricas utilizando operações booleanas de acordo com a necessidade do autor. Nesse caso, foi adicionado o elemento ressonador, a linha de alimentação, o substrato dielétrico,o plano terra, condutores internos e externos, além da porta de alimentação. As características de construção mencionadas são mostradas na Figura 4.3 e a geometria final para a simulação é mostrada na Figura 4.4.

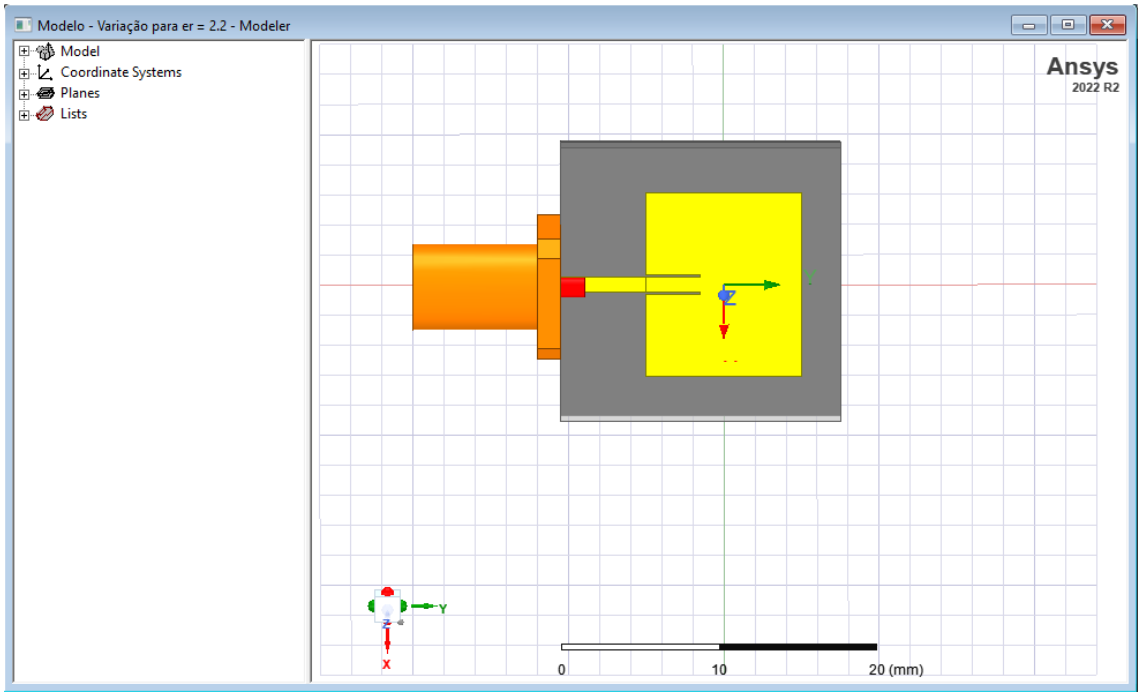
Figura 4.3 – Carcterísticas do projeto da antena.



Fonte: o autor.

A próxima etapa do modelo envolveu a parametrização da geometria da antena, ou seja, as dimensões como comprimento, largura, espessura do dielétrico e o restante das dimensões foram definidas de forma a permitir o escalonamento do projeto. Esse

Figura 4.4 – Geometria da antena.



Fonte: o autor.

escalonamento é o que possibilita a simulação de uma amplitude maior de antenas, já que ao invés de definir dimensões fixas, é definido dimensões que variam, e também dimensões que estão em função de outra dimensão variável, sendo que todo o equacionamento dessas dimensões é mostrado na Figura 4.5.

Figura 4.5 – Equacionamento das dimensões do projeto.

Properties

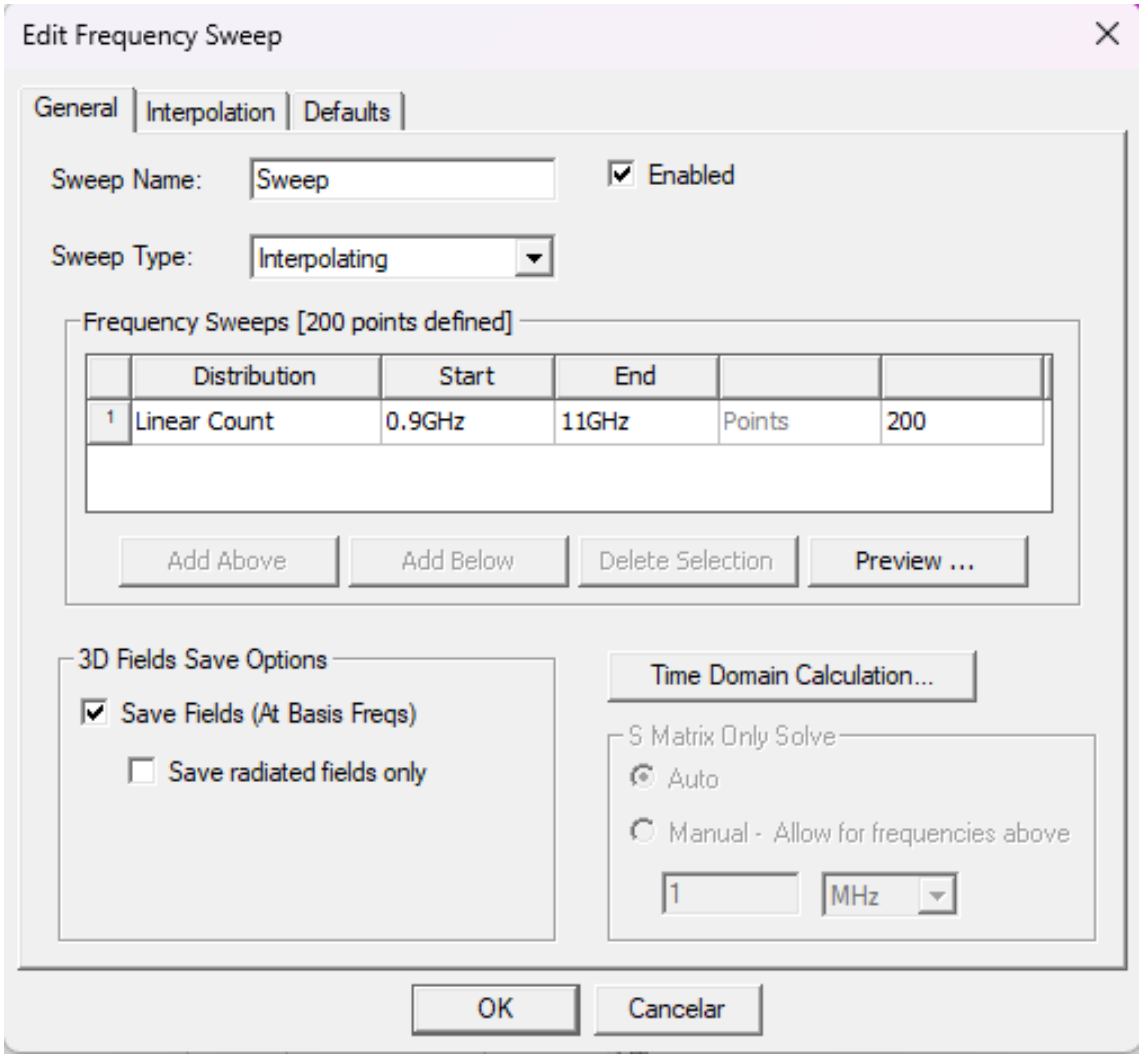
Name	Value	Unit	Evaluated Value	Type
t	2.8	um	2.8um	Design
h	1.575	mm	1.575mm	Design
Ws	1.5*W		18.27mm	Design
Ls	Ll+L+L/4		18mm	Design
W	1.218*L		12.18mm	Design
L	10	mm	10mm	Design
Wl	0.1*L		1mm	Design
Ll	0.55*L		5.5mm	Design
Wc	0.15*Wl		0.15mm	Design
Lc	0.35*L		3.5mm	Design

Fonte: o autor.

Para abranger ainda mais dados para o trabalho, foram utilizados cinco tipos diferentes de dielétricos existentes, assim tendo cinco valores de ϵ_r, mantendo a frequência de ressonância num intervalo de 0,9 GHz até 11 GHz. Além disso, a dimensão principal

L foi variada no que é denominado de *sweeps*, começando em 10 mm, com passos de 1 mm indo até 40 mm, resultando em 155 valores diferentes de frequência de ressonância a serem obtidos. Essa configuração é mostrada na Figura 4.6.

Figura 4.6 – *Sweep* de frequência do projeto.

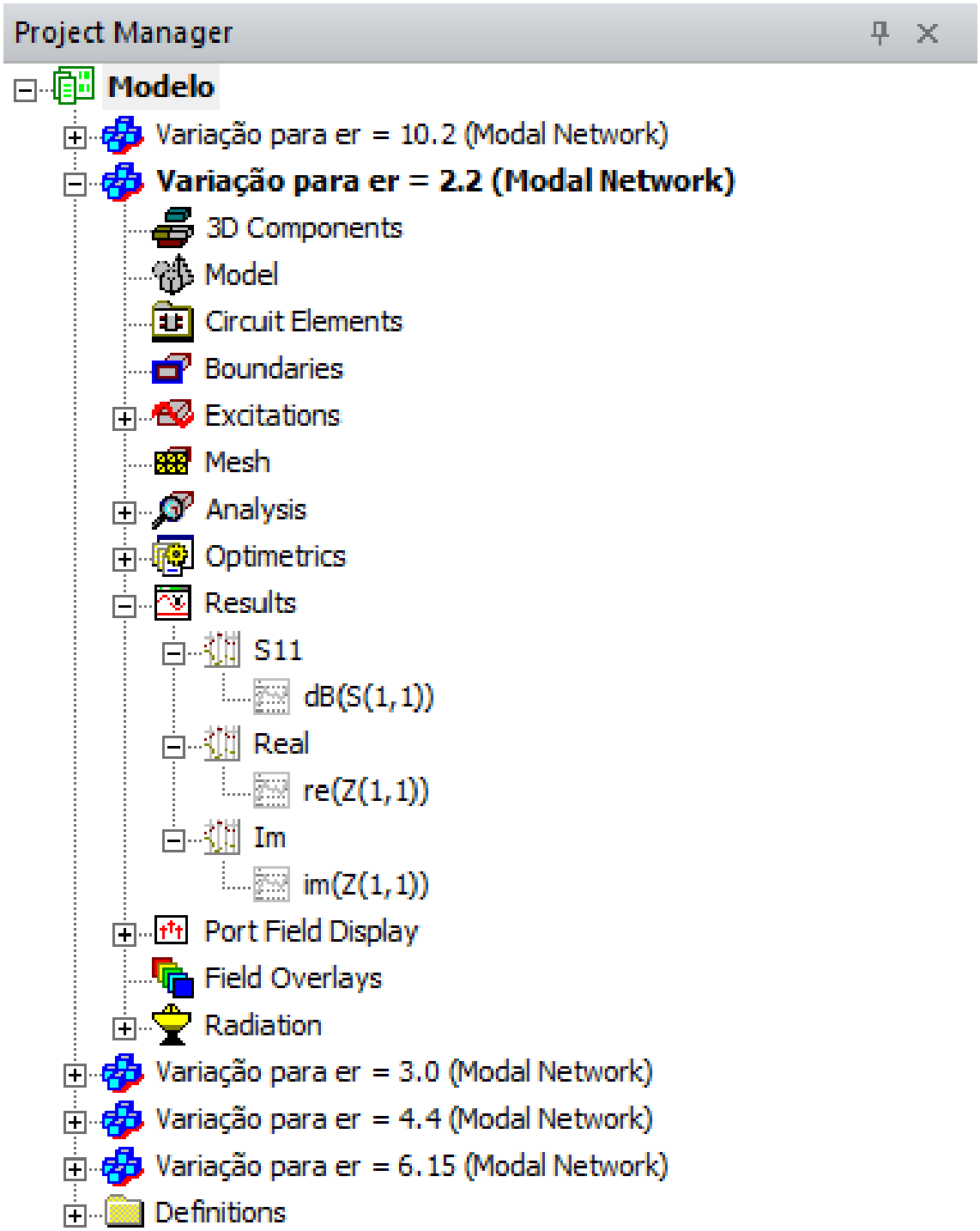


Fonte: o autor.

Com tudo isso configurado, o último passo é ir na aba do *Project Manager* tal como da Figura 4.7 onde o modelo se encontra, e inicializar a simulação dos dados, que ao terminar, fornece os resultados em parâmetros S, sendo que nesse caso o projeto possui apenas o valor de S(1,1) já que há apenas uma porta, e também em parâmetros Z para partes reais e imaginárias, tanto na forma de *plots* gráficos, tais como mostrados na Figura 4.8, Figura 4.9 e Figura 4.10, respectivamente, como também em *Datasets* no formato csv contendo os dados desse *plot*, e que é mais útil para a próxima etapa, que necessita de uma análise aprofundada desses dados.

É importante salientar que, nas simulações executadas, existem valores de resso-

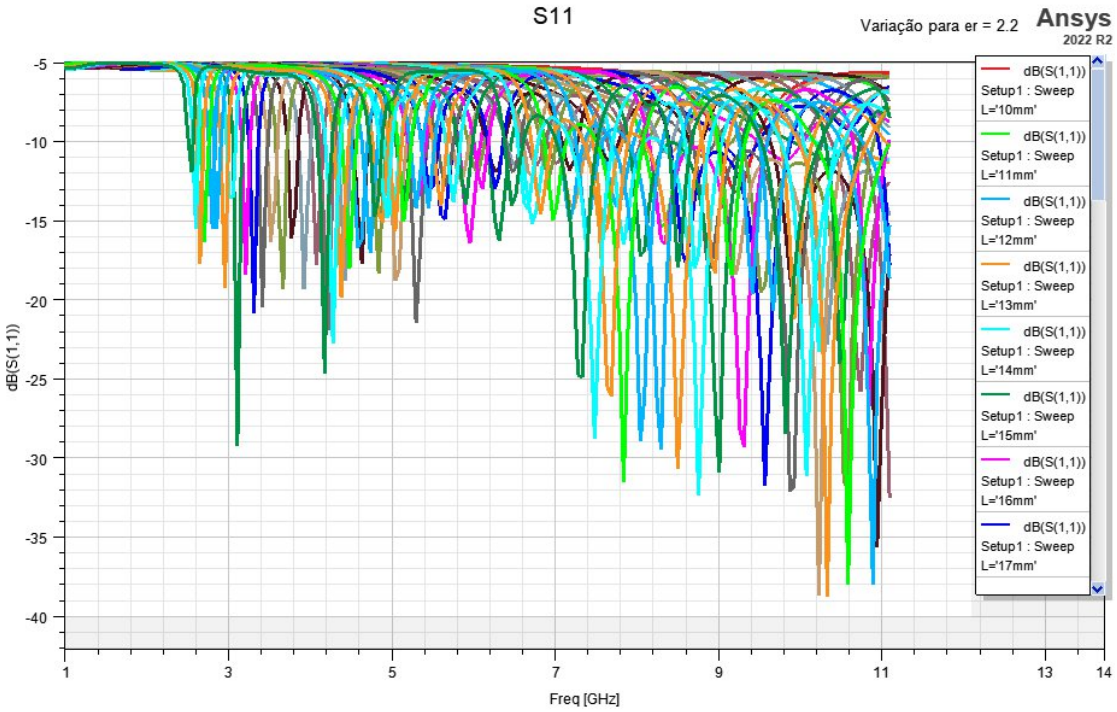
Figura 4.7 – Menu de configuração do projeto.



Fonte: o autor.

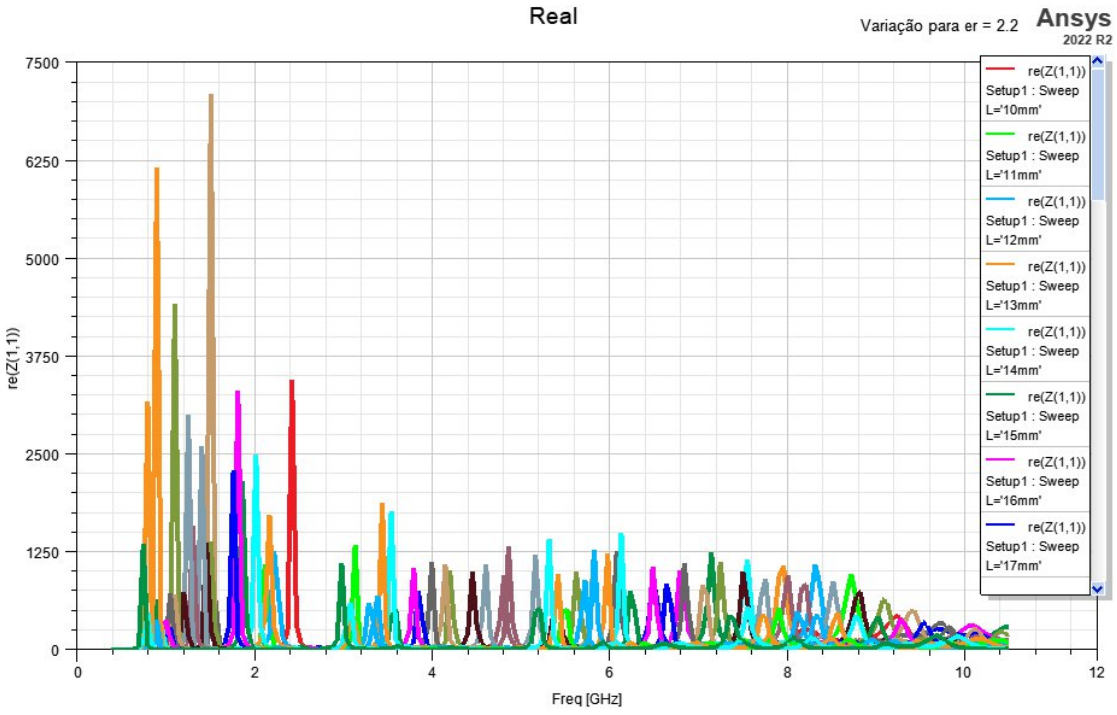
nância de outros modos ressonantes além do modo fundamental, que é o modo utilizado durante todo o projeto. Por conta disso, foi necessário empregar uma filtragem customizada nos dados, além de uma limpeza e tratamento geral antes de ser possível empregar qualquer algoritmo de *Machine Learning*, processo que será descrito na próxima seção.

Figura 4.8 – Valores de S(1,1) do projeto com $\epsilon_r = 2{,}2$.



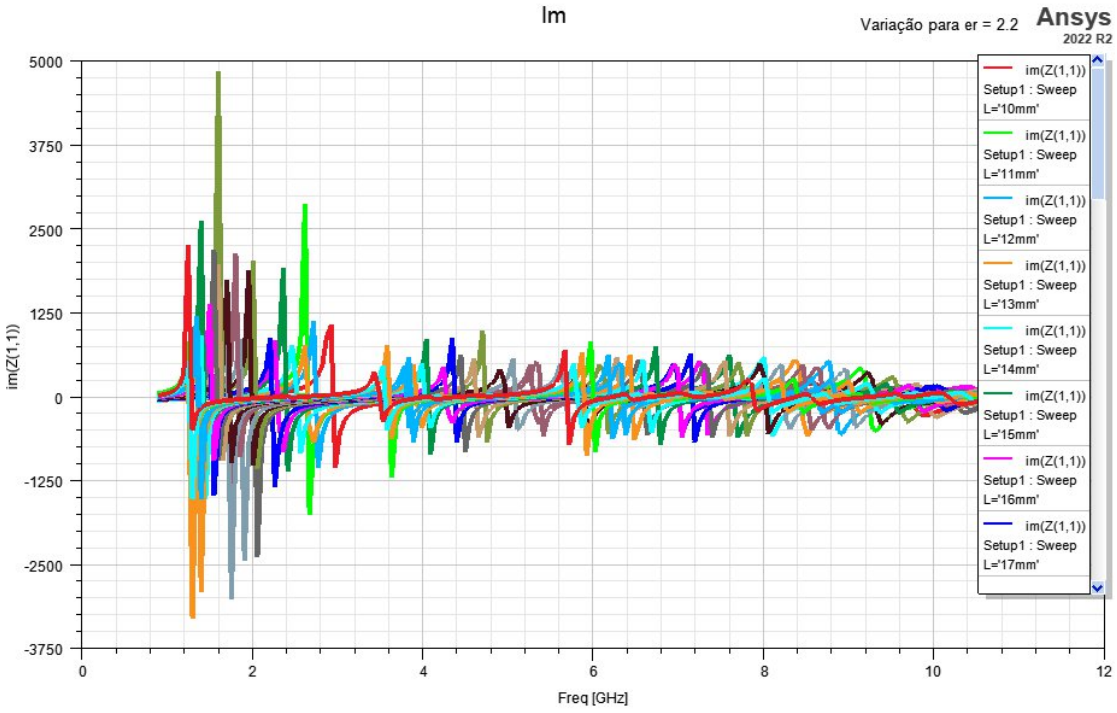
Fonte: o autor.

Figura 4.9 – Parte real do plot com $\epsilon_r = 2{,}2$.



Fonte: o autor.

Figura 4.10 – Parte imaginária do plot com $\epsilon_r = 2{,}2$.



Fonte: o autor.

4.2 Processo de Limpeza e Tratamento de Dados (*EDA - Exploratory Data Analysis*)

A partir da etapa anterior, foi possível utilizar os dados gerados para realizar uma análise e limpeza mais profunda desses dados, processo de suma importância para manter uma boa qualidade no modelo já que são eliminados dados nulos, preenchimento ou exclusão de linhas com dados faltantes e outras correções.

Toda a etapa de EDA foi feita utilizando-se o *Google Colaboratory*, que por ser toda implementada em nuvem, permite um desenvolvimento mais simples, fluido e sem que seja necessário um computador de maior poder de processamento para programação.

Dito isto, a primeira coisa feita nessa parte foi a instalação e importação das bibliotecas utilizadas no ambiente do Colab tal como no Código 4.1.

```
1 # Importando as bibliotecas Numpy, Matplotlib.pyplot, Panads e Seaborn
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import pandas as pd
5 import seaborn as sns
```

Código 4.1 – Importação de Bibliotecas

Para que não seja necessário o *upload* dos *Datasets* toda vez que uma nova sessão no Colab é inicializada, esses arquivos foram armazenados num diretório pessoal no *Google Drive*, permitindo uma conexão direta com o ambiente e sendo necessário apenas os comandos mostrados no Código 4.2 para que esses arquivos sejam baixados diretamente no ambiente de trabalho:

```
1 !gdown 1g0h18KUiMvX77avnX_9dK_U0jgy_mn3B # Real_2,2
2 !gdown 1dhVoEx7ZBpvSj-rrAh4X9fO8Ja7ogUOl # Real_3,0
3 !gdown 1p0k9vfatrCmUOsef5ELYe2ms85iPAW8H # Real_4,4
4 !gdown 1SEj43t9AWzAY6hHaRWIX4K3P3dEsaF2I # Real_6,15
5 !gdown 10EwYwMl9YvPbHZu6wRWIhNH3Hmlrlr-e # Real_10,2
```

Código 4.2 – *Download* dos *Datasets*

Como mencionado no capítulo 3, existe uma equação teórica que aproxima os resultados para a frequência de ressonância do modo fundamental das antenas de microfita. Entretanto, obtemos resultados mais confiáveis com as simulações, pois se leva em con-

sideração diversos fatores não considerados na solução analítica aproximada resolvendo o problema através de métodos numéricos.

No entanto, a solução analítica aproximada será importante em análises posteriores, e por isso sua equação é implementada já no início do programa para todos diferentes valores de L e ϵ_r, onde foi importado a constante para velocidade da luz c e também a função que calcula a raiz quadrada das bibliotecas *Scipy* e *Math*, respectivamente, como visto no Código 4.3.

```
from math import sqrt
import numpy as np
from scipy.constants import c

# List Comprehension criando valores para L variando de 10 mm ate 40 mm
# com passo de 1 mm
lista_l = [float(valor) for valor in range(10,41)]

# Funcao que gera as frequencias de ressonancia calculadas
def gerar_frequencias(lista_freq:list, eps):
    fr_0_GHz = [(c/(2 * (elem/1000)* sqrt(eps))/(10**9)) for elem in
    lista_freq]
    return fr_0_GHz

# Chamando a funcao gerar_frequencias para mapear as frequencias de
   ressonancia
# de acordo com o epsilon desejado
lista_freq_epsilon_2_20 = gerar_frequencias(lista_l, eps=2.2)

# Print estetico para separacao de linhas
print('\-/-\-/'*100)
print("\n")

# Print das listas contendo todos os valores de L e todas as frequencias
   de
# ressonancia calculadas para os diferentes valores de L e de epsilon_r
print(f'Calculo de valores da frequencia de ressonancia para todos os
```

```
valores de epsilon:')

print(f"Valores de L[mm]: {lista_l}")
print(f"Frequencias de ressonancia para epsilon =  02.20 [GHz]:
{lista_freq_epsilon_2_20}")

# Print estetico para separacao de linhas
print("\n")
print('\-/-\-/'*100)
```

Código 4.3 – Geração de listas contendo as frequências de ressonância calculadas para cada valor de L

Em seguida, foi implementado o código do Código 4.4 para leitura dos arquivos csv contendo os valores simulados, renomeando as colunas já existentes para nomes mais pertinentes, e adicionando uma nova coluna para h (espessura do dielétrico) e também uma coluna para L adicional. Essa coluna será do tipo *str* ao invés de *int*, como era originalmente, para que seja possível utilizar a técnica de *query* na criação de múltiplos *Subdatasets* organizados por L e adicionando-os em dicionários cujas chaves são seus nomes e seus valores são os *Datasets* em si. A leitura dos outros 4 *Datasets* seguem o mesmo padrão.

```
# Lendo os arquivos
arq1 = 'Real_2_2'
df1 = pd.read_csv(arq1 +'.csv')

# Renomeando as colunas originais
df1.rename(
    columns={"L [mm]": "L", "Freq [GHz]": "Freq", "re(Z(1,1)) []": "re"},
    inplace=True,
)

# Criando uma coluna para 'h' (espessura do dieletrico)
df1['h'] = df1['L'] * 0.0479

# Adicionando uma coluna com L como tipo str para fazer query
```

```
15 df1['L2'] = df1['L'].astype('str')
16
17 # Criacao de todos os subdatasets para os diferentes valores de L
18 for i in range(10, 41):
19     x = str(i)
20     query_criador_subdatasets1 = df1.query("L2 == @x")
21     query_criador_subdatasets1.to_csv('sub' + arq1 + '_' + x + '.csv',
22     index=False)
23
24 # Criando dicionarios que separam os subdatasets por epsilon_r
25 dict_2_2 = {}
26
27 for i in range(10, 41):
28     dict_2_2.update({'Real_2_2' + '_' + str(i): pd.read_csv('sub' +
29     'Real_2_2' + '_' + str(i) + '.csv')})
```

Código 4.4 – Leitura dos *Datasets*

Para mostrar que os *subdatasets* foram criados e também atualizados, podemos utilizar a função da biblioteca *Pandas, info()*, ou, *dtypes* para obter informações gerais dos dados contidos no *Dataframe*, como visto na Figura 4.11 e *head()* ou *tail()* para visualizar as primeiras, ou últimas linhas, respectivamente. Um exemplo é mostrado no Código 4.5.

```
1 # Informacoes sobre o dataset 'dict_2_2['Real_2_2_10']'
2 dict_2_2['Real_2_2_10'].info()
3
4 dict_2_2['Real_2_2_10'].tail(50)
```

Código 4.5 – Informações de um *Dataset*

Na próxima etapa, é feita a limpeza dos modos não fundamentais contidos nos *Datasets*. Para que isso seja feito, foi necessário a utilização de um raciocínio mais elaborado dentro desses dados.

Nessa análise, foi utilizada a parte real das simulações, mas algo parecido também poderia ter sido feito utilizando as partes imaginárias ou S(1,1). Entretanto, como a parte real envolve apenas valores positivos e não depende de casamento de impedância, ela foi escolhida a fim de simplificação do processo.

Figura 4.11 – Informações sobre os dados do *Dataset*.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   L       200 non-null    int64
 1   Freq    200 non-null    float64
 2   re      200 non-null    float64
 3   h       200 non-null    float64
 4   L2      200 non-null    int64
dtypes: float64(3), int64(2)
memory usage: 7.9 KB
```

Fonte: o autor.

Figura 4.12 – Saída no processo de análise dos picos do *Dataset*.

	L	Freq	re	h	L2	max_re	Maior_que_anterior	Pico_re
150	40	8.513065	30.888515	1.916	40	30.888515	True	False
151	40	8.563819	13.470976	1.916	40	13.470976	False	False
152	40	8.614573	7.128243	1.916	40	7.128243	False	False
153	40	8.665327	5.574086	1.916	40	5.574086	False	False
154	40	8.716080	5.731526	1.916	40	5.731526	True	False
155	40	8.766834	8.441736	1.916	40	8.441736	True	False
156	40	8.817588	20.467908	1.916	40	20.467908	True	False
157	40	8.868342	33.882560	1.916	40	33.882560	True	False

Fonte: o autor.

Para retirada das frequências de ressonância de modos não fundamentais dos dados, devemos analisar todos os picos contidos dentro desses *Datasets*, e esses picos são sempre maiores que o valor anterior. Para isso, foi criada uma nova coluna denominada 'maior que anterior' que avalia apenas se o valor atual é maior que o anterior. Já para saber que se trata de fato de um pico e não de um valor que seja apenas maior que o anterior, avalia-se se o ponto é maior que ambos valores, precedente e subsequente dentro da coluna 'maior que anterior', além de inicializar uma nova coluna 'pico re', que será responsável por conter apenas as linhas que de fato são picos como *True*. A Figura 4.12

mostra o processo descrito em um dos *Datasets*.

Para que fosse possível efetuar esta análise, recorreu-se a uma técnica denominada de *rolling window* que é bem semelhante a um filtro de média móvel e que avalia valores dentro de uma janela para constatação de picos, sendo essa janela de qualquer tamanho necessário. Para esse problema, como foram avaliados pares de pontos subsequentes, o tamanho da janela é de 1 unidade.

Os códigos desenvolvidos para a etapa anterior são tais como apresentado no Código 4.6 e mostra o exemplo para 1 dos 5 *Datasets*:

```
window_size = 1

for i in range(10, 41):
    dict_2_2['Real_2_2' + '_' + str(i)]['max_re'] = dict_2_2['Real_2_2' +
    '_' + str(i)]['re'].rolling(window_size, min_periods=0).max()

    dict_2_2['Real_2_2' + '_' + str(i)]['Maior_que_anterior'] = (dict_2_2
    ['Real_2_2' + '_' + str(i)]['re'] == dict_2_2['Real_2_2' + '_' + str(i
    )]['max_re']) & (dict_2_2['Real_2_2' + '_' + str(i)]['re'] > dict_2_2[
    'Real_2_2' + '_' + str(i)]['re'].shift())

    dict_2_2['Real_2_2' + '_' + str(i)]['Pico_re'] = False


# Analise linha a linha da coluna 'Maior_que_anterior' procurando por
# valores True que sejam precedidos de outros valores True e que tenham
    um valor False posterior
for i in range(0,199):
    for j in range(10,41):
        dict_2_2['Real_2_2' + '_' + str(j)].iloc[i, 7] = dict_2_2[
        'Real_2_2' + '_' + str(j)].iloc[i,6] and ~ dict_2_2['Real_2_2' +
        '_' + str(j)].iloc[i+1,6]
```

Código 4.6 – Código para encontrar todos os picos nos *Datasets*

A seguir, para obter-se apenas frequências de ressonância do modo fundamental, utiliza-se os valores obtidos pela equação aproximada implementada no Código 4.3,

utilizando o raciocínio a seguir.

O ponto de frequência simulado do modo fundamental é aquele que apresenta valor *True* na coluna 'pico re' e que possui menor diferença absoluta entre a frequência simulada e a frequência calculada. O resultado da operação de diferença absoluta mencionada é então armazenado numa nova coluna denomianda 'Diferenca fr calculado simulado' como é demonstrado na exemplificação de um dos *Datasets* mostrado na Figura 4.13.

Figura 4.13 – Saída de um dos *Datasets* com valores atualizados.

	L	Freq	re	h	L2	max_re	Maior_que_anterior	Pico_re	Diferenca_fr_calculado_simulado
40	10	2.930151	3434.795394	0.479	10	3434.795394	True	True	7.175851
154	10	8.716080	263.893462	0.479	10	263.893462	True	True	1.389921
174	10	9.731156	428.707557	0.479	10	428.707557	True	True	0.374846

Fonte: o autor.

Dessa forma, a última coisa a se fazer é separar o ponto de frequência de ressonância que satisfazem as condições especificadas anteriormente, implementado como um filtro similar ao mostrado no Código 4.7.

```
novo_teste_dataframe[((
        novo_teste_dataframe['Diferenca_fr_calculado_simulado'] ==
        novo_teste_dataframe['Diferenca_fr_calculado_simulado'].min()) &
        (novo_teste_dataframe['Pico_re'] == True))]
```

Código 4.7 – Código para encontrar a frequência de ressonância do modo fundamental

A última parte da EDA envolve, então, a criação de *Datasets* que contenham apenas os valores de frequência de ressonância do modo fundamental separados por ϵ. Para isso, primeiramente eles são construídos e inicializados sem nenhum valor, como implementado no Código 4.8 para um *Dataset*.

```
dataframe_antenas_final_2_2 = pd.DataFrame({'L':[0],
                                    'Freq':[0],
                                    're':[0],
                                    'h':[0],
                                    'L2':0,
                                    'max_re':[0],
                                    'Maior_que_anterior':[True],
                                    'Pico_re':[True],
```

```
                                    'Diferenca_fr_calculado_simulado':[0]},
                                    index=np.arange(31))
```

Código 4.8 – Código para criação de *Datasets* vazios

Finalmente, utiliza-se o filtro do Código 4.7 iterando por todos os *Dataframes* para obter todas as 31 diferentes frequências de ressonância por ϵ, adicionando-as aos novos *Datasets* e assim separando todos os valores que não serão utilizados nas etapas posteriores. Toda essa implementação, além do salvamento dos dados em novos arquivos do tipo csv é mostrada no Código 4.9.

```
for i in range(0,31):
    filtro_final = dict_2_2['Real_2_2_' + str(i+10)].query('Pico_re ==
    True').copy()

    filtro_final = filtro_final[((
    filtro_final['Diferenca_fr_calculado_simulado'] ==
    filtro_final['Diferenca_fr_calculado_simulado'].min()) &
    (filtro_final['Pico_re'] == True))].copy()

    filtro_final.reset_index(inplace=True, drop=True)

    dataframe_antenas_final_2_2.iloc[i] = filtro_final.copy()

# Salvamento dos dados somente de frequencias de ressonancia de modos
   fundamentais

dataframe_antenas_final_2_2.to_csv('dataframe_antenas_final_2_2.csv',
index=False)
```

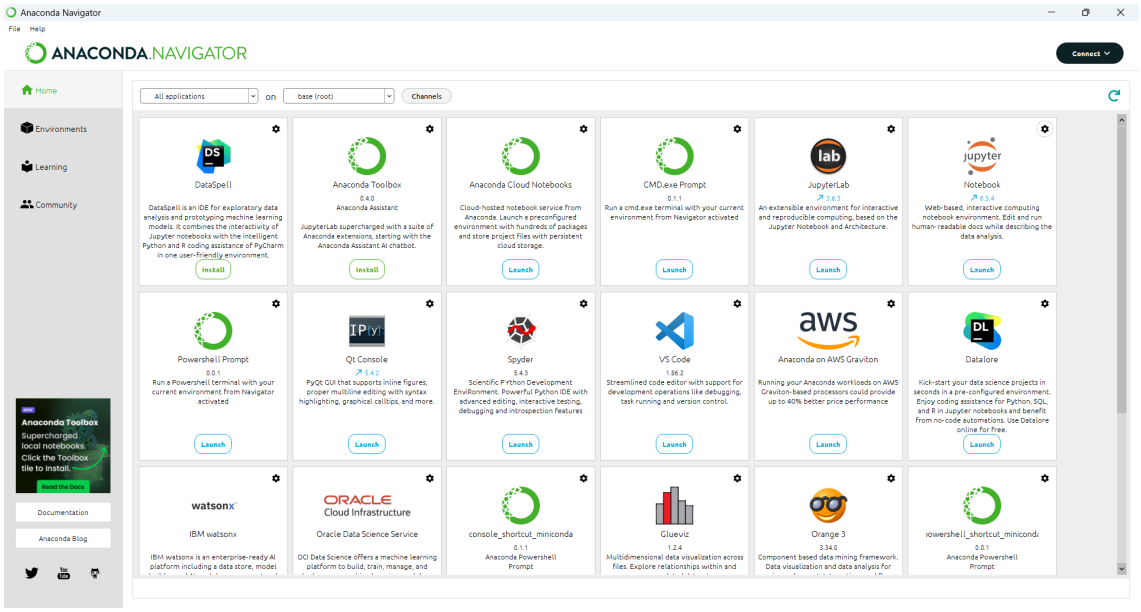
Código 4.9 – Código para adição dos pontos desejados aos novos *Dataframes*

Um processo semelhante foi desenvolvido para um segundo conjunto de dados onde o *range* de frequência foi mantido fixo de 1 GHz até 10 GHz, então os valores de L são simulados de acordo com o intervalo de frequência e o valor de ϵ.

4.3 Treinamento do Modelo de *SVR*

Para a etapa final do projeto, ao invés da ferramenta *Google Colaboratory*, foi utilizado o *Anaconda Navigator* com um *Jupyter Notebook*, ferramenta esta que torna o treinamento de modelos e qualquer outra prática de *Data Science* muito mais rápida e fluída. Podemos ver a tela inicial do *Anaconda Navigator* na Figura 4.14.

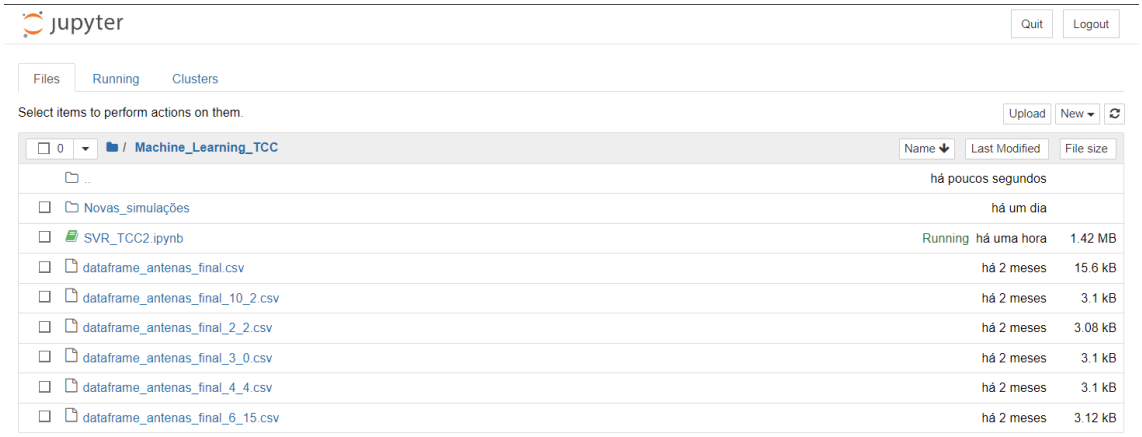
Figura 4.14 – Tela inicial do Anaconda Navigator.



Fonte: o autor.

O primeiro passo nessa etapa, é obter os *Datasets* gerados na etapa de EDA e colocar num local adequado para suas leituras, preferencialmente na mesma pasta onde esse *Jupyter Notebook* se encontra. Um exemplo é mostrado na Figura 4.15.

Figura 4.15 – Exemplo de armazenamento de *Datasets*.



Fonte: o autor.

Com tudo configurado, a próxima etapa envolve a importação das bibliotecas necessárias, sendo elas *Numpy* para computação númerica, *Matplotlib* e *seaborn* para plots gráficos *Scikit-Learn* para preprocessamento de dados e *Machine Learning*, *Pandas* para manipulação e importação de dados.

A leitura dos 5 *Datasets* é feita através da biblioteca *Pandas*, armazenando os dados em dicionários diferentes separados pelo valor de ϵ, onde a chave é um padrão com 'df' mais um valor de 0 a 4 de acordo com ϵ, e o valor é o *Dataset* lido, utilizando o Código 4.10

```
# Lista com os valores de epsilon para leitura de arquivos
lista_epsilon = ['_2_2', '_3_0', '_4_4', '_6_15', '_10_2']

# Um dicionario cuja chave e um nome que segue um padrao para facilitar o
    acesso, e o valor sao os datasets em si

dicionario_df = {'df' + str(i): pd.read_csv('dataframe_antenas_final' +
lista_epsilon[i] + '.csv') for i in range(0,5)}
```

Código 4.10 – Leitura dos dados em novos dicionários.

Em seguida, é feita uma separação entre dados que serão utilizados no *SVR* e os que não serão utilizados. Dados que servem para entradas em modelos de *Machine Learning* são denominados de *input* ou *features* e eles são utilizados para prever um resultado, como mostra (23). Já a saída do modelo, ou seja, o valor a ser previsto, em problemas de regressão é denominado de *target* (24). Como o objetivo é prever a dimensão *L (target)* utilizando a frequência de ressonância f_r *(feature)*, foi separado um novo *Dataset* a partir do anterior somente com essas duas variáveis, donde separamos as *features* na variável "X" de acordo com o valor de ϵ e os valores *target* "y" também de acordo com ϵ, como mostra o Código 4.11.

```
# Uma list comprehension que faz a copia das colunas necessarias no
   dicionario 'dicionario_df' para todos os datasets
parameters_dataset = [dicionario_df['df' + str(i)][['L', 'Freq']].copy()
for i in range(5)]

# Fazendo reshape dos Dados para treinar o modelo de SVR
X_0 = parameters_dataset[0]['Freq'].to_numpy().reshape(-1,1)
```

```
y_0 = parameters_dataset[0]['L'].to_numpy().reshape(-1,1)

X_1 = parameters_dataset[1]['Freq'].to_numpy().reshape(-1,1)
y_1 = parameters_dataset[1]['L'].to_numpy().reshape(-1,1)

X_2 = parameters_dataset[2]['Freq'].to_numpy().reshape(-1,1)
y_2 = parameters_dataset[2]['L'].to_numpy().reshape(-1,1)

X_3 = parameters_dataset[3]['Freq'].to_numpy().reshape(-1,1)
y_3 = parameters_dataset[3]['L'].to_numpy().reshape(-1,1)

X_4 = parameters_dataset[4]['Freq'].to_numpy().reshape(-1,1)
y_4 = parameters_dataset[4]['L'].to_numpy().reshape(-1,1)
```

Código 4.11 – Separação e *reshape* para adequação dos dados *dataframes*.

Como o valor de L é variado, mas mantido no intervalo de 10 mm a 40 mm, de acordo com a equação para frequência de ressonância aproximada, as curvas de regressão serão diferentes e terão intervalos diferentes de funcionamento para a frequência de ressonância, com seus mínimos e máximos diferentes, como vistos no Código 4.12.

```
# Valores minimos e maximos dentro da coluna 'Freq' (Para os 5 datasets)
max_freq_0 = max(parameters_dataset[0]['Freq'])
min_freq_0 = min(parameters_dataset[0]['Freq'])

max_freq_1 = max(parameters_dataset[1]['Freq'])
min_freq_1 = min(parameters_dataset[1]['Freq'])

max_freq_2 = max(parameters_dataset[2]['Freq'])
min_freq_2 = min(parameters_dataset[2]['Freq'])

max_freq_3 = max(parameters_dataset[3]['Freq'])
min_freq_3 = min(parameters_dataset[3]['Freq'])

max_freq_4 = max(parameters_dataset[4]['Freq'])
min_freq_4 = min(parameters_dataset[4]['Freq'])
```

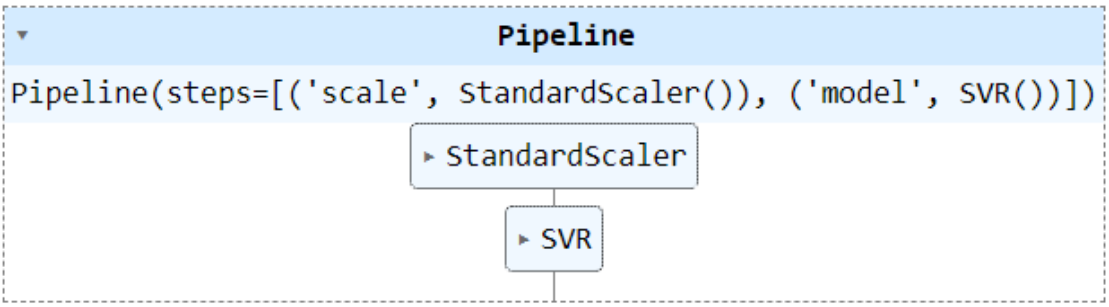
```
16
17 print(max_freq_0, min_freq_0)
18 print(max_freq_1, min_freq_1)
19 print(max_freq_2, min_freq_2)
20 print(max_freq_3, min_freq_3)
21 print(max_freq_4, min_freq_4)
```

Código 4.12 – Código para visualizar as frequências mínimas e máximas dos dados.

Para padronização do processo e facilidade de reprodutibilidade, foram utilidas as *Pipelines* do *Scikit-Learn*, que se trata da aplicação de transformações de forma sequencial e ordenada a diferentes modelos através de uma única função (25).

A única transformação em si utilizada no projeto foi a aplicação de um *Standard Scaler*, e isso se deve ao fato de que as variáveis são mensuradas em escalas diferentes e portanto contribuem de forma diferente para o treinamento, o que pode resultar na introdução de um viés no modelo (26). Além do *Standard Scaler*, é incluido também na *Pipeline* o tipo de modelo a ser utilizado, nesse caso o *SVR*. Um resumo da *Pipeline* é mostrada na Figura 4.16 e o código para sua criação está no Código 4.13.

Figura 4.16 – Resumo da *Pipeline*.



Fonte: o autor.

```
1 # Criando uma Pipeline de treinamento com StandardScaler para modelo de
    SVR
2 pipe = Pipeline([
3     ('scale', StandardScaler()),
4     ('model', SVR())
5 ])
```

Código 4.13 – Código para criação de uma *Pipeline* no sklearn.

Um aspecto importante do *Machine Learning* com aprendizado supervisionado é que o modelo utilizado consiga generalizar seu aprendizado sobre o conjunto de dados utilizados, ou seja, treina-se o modelo utilizando alguns dados e então deseja-se que o modelo realize previsões com dados nunca vistos antes, pois dessa forma é possível avaliar a performance. Se o modelo tem *overfitting*, o modelo aprende muito bem acerca de seus dados, entretanto, não consegue fazer previsões plausíveis para dados nunca vistos, ou se o modelo tem *underfitting*, o modelo não consegue aprender o suficiente sobre os dados de treinamento e efetuará previsões com erros elevados (27).

Uma das formas de contornar esse problema é a divisão dos dados em uma partição para treinamento do modelo, contendo a maior parte dos dados totais, e outra partição para realizar testes no modelo, já que nesse sentido esses dados de teste nunca foram vistos anteriormente, e então é possível realizar uma avaliação mais concisa. Foi feita a divisão para cada valor de ϵ conforme mostrado no Código 4.14, onde 80% dos dados são de treino e 20% para testes.

```
X_train_0, X_test_0, y_train_0, y_test_0 = train_test_split(X_0, y_0,
test_size=0.2, random_state=2024)

X_train_1, X_test_1, y_train_1, y_test_1 = train_test_split(X_1, y_1,
test_size=0.2, random_state=2024)

X_train_2, X_test_2, y_train_2, y_test_2 = train_test_split(X_2, y_2,
test_size=0.2, random_state=2024)

X_train_3, X_test_3, y_train_3, y_test_3 = train_test_split(X_3, y_3,
test_size=0.2, random_state=2024)

X_train_4, X_test_4, y_train_4, y_test_4 = train_test_split(X_4, y_4,
test_size=0.2, random_state=2024)
```

Código 4.14 – Partição dos dados em treino e teste.

Em seguida, uma *Pipeline* para cada valor de ϵ é treinada e armazenada numa lista para serem utilizadas posteriormente na fase de *Hyperparameter Tunning*, que consiste em encontrar os hiperparâmetros que produzem o melhor modelo possível (28), como mostrado no Código 4.15.

```
# Treinando as Pipelines
lista_pipelines = [pipe.fit(X_train_0, y_train_0),
                   pipe.fit(X_train_1, y_train_1),
                   pipe.fit(X_train_2, y_train_2),
                   pipe.fit(X_train_3, y_train_3),
                   pipe.fit(X_train_4, y_train_4)]
```

Código 4.15 – Treinamento das *Pipelines*

Para melhor aproveitamento da etapa de *Hyperparameter Tuning*, é possível visualizar todas as variáveis livres, ou seja, as variáveis do modelo que podem sera alteradas utilizando a função da própria *Pipeline* denominada *get params()*, que retorna um dicionário cuja chave é o nome do parâmetro e o valor é o seu valor padrão, como mostrado no Código 4.16.

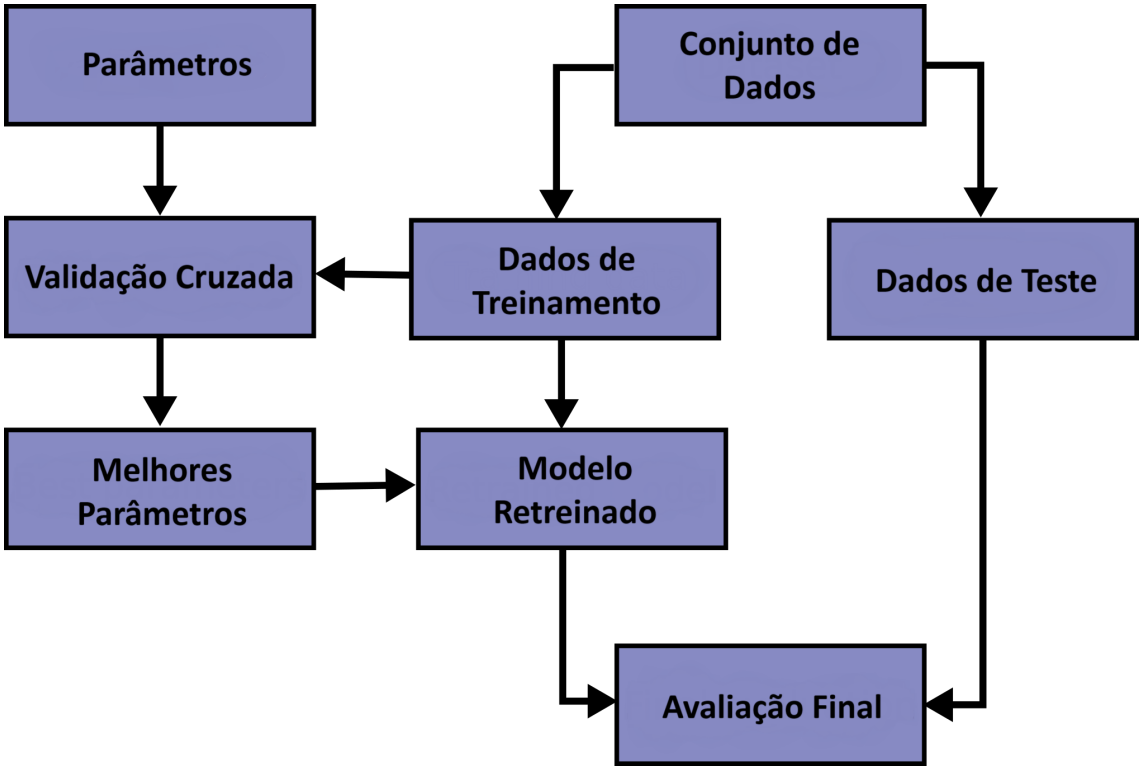
```
# Informacoes sobre todos os parametros das Pipelines
for i in range(len(lista_pipelines)):
    print(f"Pipeline do dataset: {'parameters_dataset_' + str(i)}")
    print(f'--' *60)
    for k, v in lista_pipelines[i].get_params().items():
        print(f'Key: {k} --> Value: {v}')

    print(f'--' *60)
    print(f'\n\n')
```

Código 4.16 – Parametros do Modelo

Quando se particiona os dados em treinamento e teste, há uma perda nos dados que de outra forma, seriam utilizados inteiramente para treinamento. Essa redução nas amostras de treinamento implica que, para que o treinamento seja bem feito, amostras aleatórias para pares de teste e validação que caracterizem bem os dados são necessárias, e não é possível contar com essas aleatoriedades ao construir modelos de *Machine Learning* (29). Uma estrutura corriqueira de treinamento de modelo, com a inclusão de validação cruzada (*cross validation*) é vista na Figura 4.17.

A forma mais básica de validação cruzada é denominada de *k-fold*, que consiste na divisão dos dados em k partes com aproximadamente a mesma quantidade de amostras e faz o treinamento e teste do modelo k vezes, utilizando na primeira vez uma das k partes

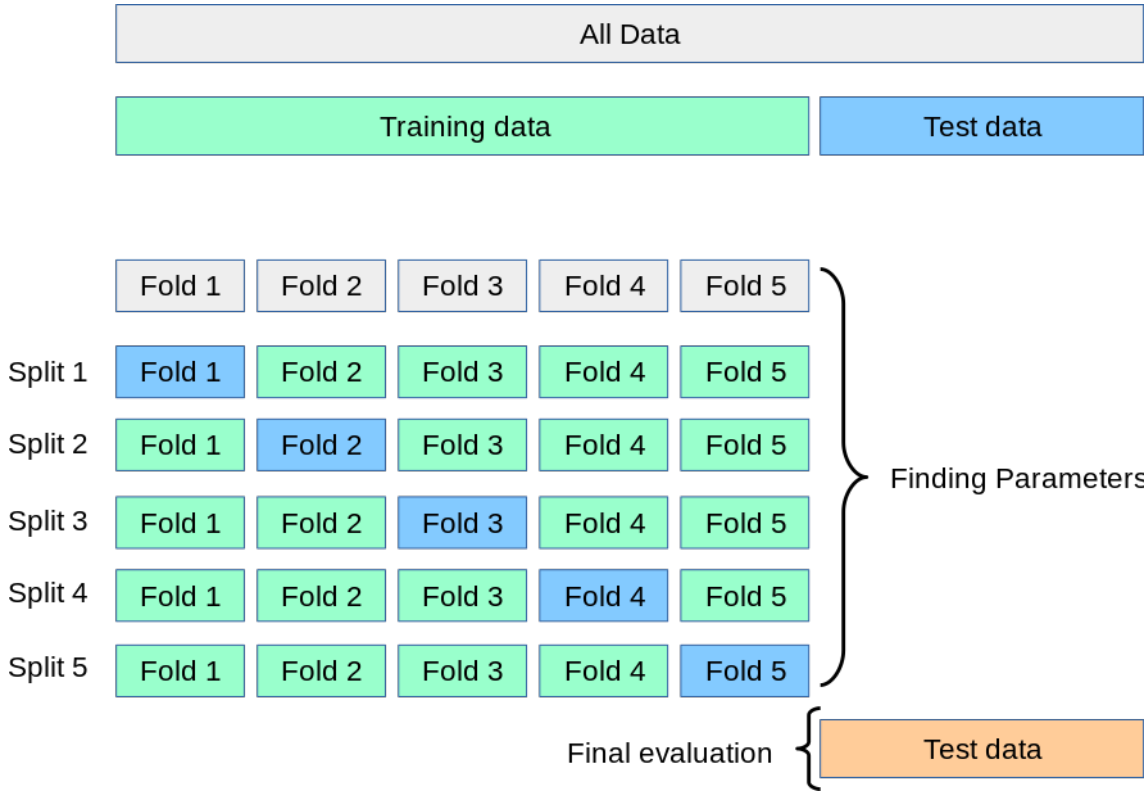
Figura 4.17 – Fluxograma de Treinamento.



Fonte: Adaptado de (29).

como teste e o restante para validação, na próxima vez outra das k partes como teste e o restante para validação, e assim por diante. A Figura 4.18 mostra esse processo.

Figura 4.18 – Funcionamento do *k-fold*.



Fonte: (29).

Nesse projeto foi implementado uma *GridSearchCV*, que é similar ao *k-fold*, mas que além de efetuar a validação cruzada, permite aplicar o *Hyperparameter Tuning*, ou seja, além de validar os dados, podemos também validar a melhor combinação possível de hiperparâmetros passados ao nosso modelo simplesmente passando uma *param grid* que contém todos os valores desejados para os hiperparamêtros a serem testados, como visto para uma das *GridSearchCV* no Código 4.17. Um ponto negativo evidente dessa técnica é que o *tuning* é feito somente com valores que passamos para a *GridSearchCV*, podendo não ser no fim das contas o melhor modelo possível, além de gastar muito poder computacional.

```
1 # Parametros para as GridSearch
2 param_grid={'model__C': [10, 100, 150, 200, 400, 750, 1000, 1500, 5000,
3                          10000, 50000],
4              'model__kernel':['rbf'],
5              'model__epsilon':[0.01, 0.05, 0.09, 0.1, 0.2, 0.3, 0.4, 0.5,
6              0.6, 0.7, 0.8, 0.9]
7              }
```

```
8
9  scoring = 'r2'
10 cv = 3
11
12 # 5 grids, 1 para cada dataset utilizando cada uma uma pipeline diferente
       da lista 'lista_pipelines'
13 grid_0 = GridSearchCV(estimator=lista_pipelines[0],
14                        param_grid=param_grid,
15                        scoring=scoring,
16                        cv=cv
17                       )
```

Código 4.17 – Código para criar os parâmetros da GridSearchCV

Por fim, a única coisa necessária a se fazer é treinar todas as *GridSearch* construídas com os dados anteriormente separados para treinamento, obtendo os modelos de SVR treinados, utilizando um simples comando *.fit()*, como visto no Código 4.18

```
1 # Treinamento de todas as grids
2 grid_0.fit(X_train_0, y_train_0)
3 grid_1.fit(X_train_1, y_train_1)
4 grid_2.fit(X_train_2, y_train_2)
5 grid_3.fit(X_train_3, y_train_3)
6 grid_4.fit(X_train_4, y_train_4)
```

Código 4.18 – Código para Treinar as GridSearchCV

Assim como na etapa de EDA, o mesmo processo foi repetido para outro *Dataset* onde a frequência na simulação foi fixada entre 1 GHz e 10 GHz.

5 Resultados e Discussões

5.1 Considerações Iniciais

Neste capítulo são apresentados os resultados obtidos para o processo de SVR considerando alguns fatores. Para melhores comparações de resultados, foram simulados 2 tipos diferentes de *Datasets*. No primeiro tipo de *Dataset*, os valores de L ficaram fixos entre 10 e 40 mm, variando com passo de 1 mm. Já no outro *Dataset*, a frequência de ressonância foi mantida sempre entre 1 GHz e 10 GHz, e por conta disso, os valores de L não são os mesmos para todos os valores de ϵ. Além disso, foram implementados modelos com e sem a utilização de *train/test split*, a fim de avaliar como a separação de dados para treinamento e teste afetam a *performance* do modelo. Por fim, foram utilizadas métricas estatísticas como o *MAE - Mean Absolute Error* e o *MSE - Mean Squared Error*.

5.2 Montagem, Construção e Simulação do Modelo das Antenas

O ponto inicial do trabalho é a conciliação de duas áreas da tecnologia extremamente importantes para a sociedade contemporânea, as telecomunicações e a inteligência artificial. As IA's são muito aplicadas nas comunicações de forma geral, entratanto, suas aplicações em projetos de antenas ainda não são tão exploradas como poderiam.

A fundamentação deste estudo baseia-se nos trabalhos desenvolvidos em (9) e (8). O primeiro apresenta uma análise detalhada para a construção de antenas de microfita retangulares voltadas ao handover entre as tecnologias 4G e 5G. O segundo investiga o emprego de diversas técnicas de Machine Learning na previsão de características dessas antenas.

Com base nas parametrizações de dimensões estabelecidas no trabalho de (9) mostradas na Figura 4.5, obtém-se o equacionamento das dimensões em função exclusiva de L.

Essas equações estabelecem as proporcionalidades entre as dimensões das antenas, sendo $h = 0{,}0479L$, $W = 1{,}218L$, $L_I = 0{,}55L$, $W_s = 1{,}5W$, $Ls = L_I + L + \frac{L}{4}$, $W_I = 0{,}1L$, $W_c = 0{,}15W_I$, e por último, $L_c = 0{,}35L$.

Como fica evidente pelas proporcionalidades apresentadas, as antenas variam, em termos de dimensões físicas, única e exclusivamente de acordo com as variações da

dimensão L, o que evidencia a necessidade de se encontrar uma forma mais rápida para encontrar essa dimensão utilizando outros métodos que levem menos tempo e que não contenham tanto erro quanto a utilização de equações, por exemplo.

Todo esse trabalho caracteriza a simulação de apenas 1 antena no *software ANSYS HFSS*, assim, é necessário utilizar formas de automatizar a simulação de múltiplas antenas diferentes de uma vez só.

Sendo assim, foram utilizados os *sweeps*, que permitem configurar o intervalo de frequência a ser analisado, bem como a quantidade de pontos para convergência de resultados, que para esse caso foram utilizados 200 pontos, método para convergência, e outras configurações. Já o *Parametric Setup* permite a definição da forma como será feita a avaliação das dimensões físicas da antena, permitindo análise de 1 único ponto, pontos distintos especificados, intervalo de valores igualmente espaçados, e outras configurações.

5.3 Métricas de Avaliação do Modelo

Quando se trata de modelos de *Machine Learning*, seja ele de regressão ou de classificação, a melhor forma de avaliar se esse modelo tem uma boa performance é com a utilização da estatística.

As métricas utilizadas dependem do tipo de problema a ser solucionado, e cada uma delas é boa para elucidar acerca de determinados problemas e características do modelo. A fim de uma análise detalhada, porém compacta, apenas as principais métricas de avaliação de modelos de regressão foram utilizadas no trabalho. Inicialmente, são apresentadas suas equações e como interpretar o resultado obtido, e então é feita uma análise com todas as *GridSearchCV* criadas a fim de avaliação desses resultados de forma a englobar todas as curvas de regressão.

5.3.1 *R2 Score*

Uma das métricas mais utilizadas para avaliar os modelos e o R2 *Score*. O R2 consegue avaliar a dispersabilidade em modelos de regressão, sendo um valor entre 0 e 1. Ele indica a porcentagem da variância dos dados que podem ser explicadas pelo modelo, e assim sendo, o quanto fatores aleatórios afetam esse modelo. Dessa forma, em geral, quanto mais próximo de 1 a métrica R2 está, melhor é o modelo.

Com a biblioteca *Scikit-Learn*, a computação dessa e de diversas outras métricas é feita com apenas uma linha de comando, importando as funções do módulo *skle-*

arn.metrics, e usando 2 argumentos na função, o valor *target* de teste conhecido (y_test) e o valor predito de *target* feito pelo modelo. A equação para o *R2 Score* é mostrada na Equação 5.1 conforme a referência de (30):

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{5.1}$$

De forma que:

y_i é o valor real de cada y.

$\hat{y}_i$ é o valor previsto de cada y.

$\bar{y}$ é o valor da média dos valores de y.

5.3.2 *MAE - Mean Absolute Error*

Outra métrica importantíssima na avaliação de modelos de regressão é o MAE (*Mean Absolute Error*), que como o nome indica, demonstra qual o erro médio absoluto do modelo.

O MAE soma os erros (valor real - valor predito) e tira a média desse valor, indicando, em média, quanto é o erro de previsão do modelo.

A equação para calcular o valor do MAE é mostrado na Equação 5.2.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{5.2}$$

De modo que:

y_i é o valor real de cada i-ésimo y.

$\hat{y}_i$ é o valor previsto para cada i-ésimo y.

5.3.3 *MSE - Mean Squared Error*

O MSE *(Mean Squared Error)* é similar ao MAE, mas ao invés de apenas somar os erros e tirar a média, essa métrica eleva cada erro individual ao quadrado, soma-os, e então a média desses erros quadrados é tirada. Isso acontece porque muitas vezes, é necessário avaliar a forma como os erros afetam o modelo, e obviamente, alguns pontos produzem mais erros que outros.

O MSE consegue ver exatamente isso, já que valores pequenos de erro, elevados ao quadrado, ainda assim produzem erros pequenos, já valores grandes de erros, elevados ao quadrado, dominam o valor da métrica. Sendo assim, o MSE é excepcional para avaliar modelos onde erros não são tolerados. A Equação 5.3 caracteriza a métrica descrita.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5.3}$$

De modo que:

y_i é o valor real de cada i-ésimo y.

$\hat{y}_i$ é o valor previsto de cada i-ésimo y.

5.4 Comparação de Métricas

Dadas as métricas citadas na seção 5.3 e os resultados obtidos pela regressão versus valores reais simulados das antenas, podemos comparar, finalmente, o desempenho do algoritmo de regressão implementado no trabalho e discutir os resultados obtidos com respaldo estatístico.

Como a regressão dos dados foi feita para 5 diferentes *GridSearchCV*, que correspondem a 5 diferentes ϵ_{eff} os resultados devem ser analisados de acordo com tal fator.

5.4.1 Métricas Estatísticas

A primeira métrica analisada é o R^2Score, que para as *Grids* 1, 2, 3, e 4, o valor obtido foi de 0,99, ou seja, para essas *grids*, quase que a totalidade dos modelos empregados podem ser explicados pela variância do próprio modelo, sendo praticamente invulnerável as aleatoriedades intínsicas. Para a *Grid 0* , no entanto, o valor foi de 0,93, sendo um resultado mais ruim, o que será perceptível na análise das outras métricas.

A próxima métrica é o *MAE*, e com ela, fica evidente que, em média, o erro maior está presente na *Grid 0*($\epsilon_r = 2{,}2$), com um valor de cerca de 2,057 mm, já para as outras, o MAE foi menor que 0,35 mm, o que mostra que os maiores erros estão de fato na *Grid 0*, fato este que terão as possíveis causas de erro mais elevado discutidos posteriormente.

A terceira e última métrica foi o MSE, que como discutido na seção 5.3, tende a ascentuar de forma mais visíveis erros com valores elevados, já que ela calcula o erro, o eleva ao quadrado para cada ponto calculado e então uma média é tirada. O MSE

para a *Grid 0* teve um valor de 0.695, o que é maior que todas as outras *Grids*, que individualmente tiveram valores de MSE menor que 0,5.

A tabela 5.1 mostra os valores para todas as métricas discutidas.

ϵ_r	R^2Score	MAE	MSE
2,20	0,932	2,057	0,695
3,00	0,999	0,301	0,406
4,40	0,999	0,249	0,288
6,15	0,999	0,156	0,129
10,20	0,999	0,172	0,109

Tabela 5.1 – Comparação de métricas para as diferentes *Grids*.

5.4.2 Características e Dimensões das Antenas

Outra forma de avaliar o desempenho do modelo é a comparação direta entre os resultados das dimensões obtidas para o modelo e os valores dados pela equação. Para que não haja nenhum tipo de"viés" na escolha dos pontos escolhidos para essa comparação, foi utilizado a biblioteca *Random*, que gera um valor aleatório dentro de um intervalo especificado.

Os valores gerados pela biblioteca *Random* para os pontos de frequência foram de 3,14, 8,14, 6,32, 7,64 e 5,23 GHz, que quando substituidos na Equação 2.18, retornam os valores de L_{calc} da antena. Esses valores de L_{calc} foram então inseridos no *ANSYS* para que novas simulações sejam feitas, e então o valor real de frequência de operação $f_{simLcalc}$ seja obtido.

O mesmo processo foi feito também para o modelo de SVR, partindo das mesmas frequências iniciais, obtendo um valor de L_{SVR} predito, que foi então adicionado ao *ANSYS* para que o valor real de frequência de operação da antena $f_{simLSVR}$ seja obtida.

Foram feitas comparações para cada um dos valores de ϵ, onde é possível fazer uma melhor generalização dos resultados obtidos.

A tabela 5.2 comporta os resultados referentes ao cálculo pela equação aproximada e também pela obtenção de valores do modelo de SVR para $\epsilon = 2{,}2$. Na terceira coluna da tabela é mostrado os valores de L_{SVR} providos pelo modelo de SVR, valores estes que quando colocados para serem simulados no *ANSYS*, resultam em frequêcias de operação $f_{simLSVR}$, mostrados na quarta coluna. Ao se comparar os valores de frequência obtida para o modelo de SVR e para equação, com a frequência inicial gerada aleatoriamente (f_r), percebe-se claramente que o erro gerado para o modelo criado é con-

sistentemente menor que o erro obtido pela Equação 2.18, algo que é bem generalizado através das métricas estatísticas citadas na seção 5.4.1

ϵ_{eff}	f_r $[GHz]$	L_{SVR} $[mm]$	$f_{simLSVR}$ $[GHz]$	L_{calc} $[mm]$	$f_{simLcalc}$ $[GHz]$
2,20	3,14	31,18	3,10	32,18	3,00
	8,14	13,13	8,41	12,41	8,58
	6,32	14,55	6,37	15,99	5,82
	7,64	13,32	8,30	13,23	8,33
	5,23	18,02	5,21	19,32	4,91

Tabela 5.2 – Comparação de Resultados Para $\epsilon = 2{,}2$

Para o valor de $\epsilon = 3{,}0$, temos resultados bem semelhantes, com algumas ressalvas, já que aqui percebe-se melhor que o modelo não é perfeito e que em poucas ocasiões, o calculo pela equação fornece melhores resultados, já que no ponto de frequência $f_r = 8{,}14$ GHz, o erro é menor utilizando-se a Equação 2.18.

ϵ_{eff}	f_r $[GHz]$	L_{SVR} $[mm]$	$f_{simLSVR}$ $[GHz]$	L_{calc} $[mm]$	$f_{simLcalc}$ $[GHz]$
3,00	3,14	27,17	3,13	27,56	3,10
	8,14	10,19	10,75	10,63	7,83
	6,32	13,43	6,33	13,69	6,22
	7,64	10,82	8,06	11,33	7,33
	5,23	16,27	5,21	16,55	5,16

Tabela 5.3 – Comparação de Resultados Para $\epsilon = 3{,}0$

Para o valor de $\epsilon = 4{,}40$, o comportamento também segue o padrão anterior, onde frequências de, até aproximadamente $7GHz$, o modelo treinado consegue ceder resultados bem melhores que a Equação 2.18, conforme visto na Tabela 5.4.

ϵ_{eff}	f_r $[GHz]$	L_{SVR} $[mm]$	$f_{simLSVR}$ $[GHz]$	L_{calc} $[mm]$	$f_{simLcalc}$ $[GHz]$
4,40	3,14	22,58	3,13	22,76	3,10
	8,14	8,49	10,64	8,78	7,93
	6,32	11,21	6,33	11,31	6,27
	7,64	9,03	8,56	9,35	7,38
	5,23	13,59	5,21	13,66	5,22

Tabela 5.4 – Comparação de Resultados Para $\epsilon = 4{,}4$

Para o valor de $\epsilon = 6{,}15$, o comportamento observado anteriormente se mantém, onde o modelo tem dificuldades de prover resultados melhores para frequências maiores, conforme Tabela 5.5.

ϵ_{eff}	f_r $[GHz]$	L_{SVR} $[mm]$	$f_{simLSVR}$ $[GHz]$	L_{calc} $[mm]$	$f_{simLcalc}$ $[GHz]$
6,15	3,14	19,20	3,13	19,25	3,15
	8,14	7,22	11,09	7,42	7,98
	6,32	9,58	6,28	9,56	6,32
	7,64	7,66	8,41	7,91	7,48
	5,23	11,58	5,27	11,56	5,23

Tabela 5.5 – Comparação de Resultados Para $\epsilon = 6{,}15$

Finalmente, para o valor de $\epsilon = 10{,}20$, as conclusões também são bem parecidas quanto aos resultados, entretanto, é perceptível que ao se aumentar o valor de ϵ, o modelo treinado tende a obter melhores resultados para frequências maiores, próximas aos 9 ou até $10GHz$.

ϵ_{eff}	f_r $[GHz]$	L_{SVR} $[mm]$	$f_{simLSVR}$ $[GHz]$	L_{calc} $[mm]$	$f_{simLcalc}$ $[GHz]$
10,2	3,14	15,24	3,13	14,95	3,15
	8,14	5,67	7,95	5,76	8,03
	6,32	7,51	6,33	7,43	6,37
	7,64	6,03	7,65	6,14	7,53
	5,23	9,06	5,26	8,97	5,32

Tabela 5.6 – Comparação de Resultados Para $\epsilon = 10{,}2$

Os maiores erros aparecem em frequências de cerca de 6-8 GHz, onde uma possível causa para isso é a própria simulação do *ANSYS*, sendo que, nesse intervalo, os valores de comprimento L não parecem seguir um formato aparente.

6 Conclusão

O presente trabalho propôs a utilização de um modelo de inteligência artificial, especificamente o *Support Vector Regression (SVR)* para a criação rápida e eficiente de projetos de antenas de microfita retangulares, bastando definir o material de substrato, e consequentemente o valor de ϵ_r, além da frequência de ressonância que a antena irá operar, onde o modelo estimará as dimensões físicas (comprimento L) para o modo TM_{10}.

Para alcançar esse propósito, construiu-se uma base de dados própria originada de simulações numéricas, conduzidas no software *ANSYS Electronics Desktop 2022 (módulo HFSS)*. Foram simuladas centenas de variações dimensionais operando de 1 GHz a 10 GHz para cinco valores distintos de permissividade do dielétrico (ϵ_r = 2,2, 3,0, 4,4, 6,15 *e* 10,2), garantindo a obtenção de parâmetros S e Z condizentes com as simulações propostas.

A extração das características (EDA) conduzida em linguagem *Python*, com suporte da biblioteca Pandas, permitiu separar eficientemente os modos não fundamentais através da identificação de picos de ressonância (uso de *rolling windows*) e comparativos com a aproximação analítica teórica. Estes dados refinados alimentaram as *pipelines* do *Scikit-Learn*. A otimização dos modelos por validação cruzada ($GridSearchCV$) provou-se adequada para calibrar os hiperparâmetros de penalização (C), margem (ϵ) e seleção do *kernel* correto para o *SVR*, prevenindo que o sistema sofresse *underfitting* ou *overfitting* generalizado.

Os resultados obtidos evidenciam a boa eficácia do algoritmo de regressão proposto. A avaliação estatística confirmou uma boa capacidade preditiva, atingindo um coeficiente de determinação (R^2Score) de 0,999 e um Erro Médio Absoluto (MAE) inferior a 0,35 mm para os cenários com $\epsilon_r \geq$ 3,0. Embora o desempenho do modelo em substratos de ϵ_r = 2,2 tenha revelado um desvio ligeiramente maior (R^2 = 0,932 e MAE de cerca de 2,057 mm), o modelo se mostrou utilizável em boa parte das situações.

Mais importante ainda, o comparativo direto documentado evidenciou que a dimensão predita pela regressão (L_{SVR}) resulta em ressonâncias substancialmente mais fiéis às desejadas do que as alcançadas pelas dimensões oriundas do cálculo das equações analíticas aproximadas (L_{calc}), especialmente em frequências de ressonância inferiores a 7-10 GHz. O modelo SVR demonstrou uma habilidade excelente de incorporar efeitos de franja e não linearidades do dielétrico que a matemática tradicional desconsidera nos cálculos simples.

Conclui-se, portanto, que a integração do Aprendizado de Máquina no *design* de antenas constitui uma alternativa veloz e com boa exatidão, minimizando o pesado custo computacional e humano gasto nos ajustes empíricos durante múltiplas simulações de projeto.

Como sugestão para propostas de trabalhos futuros, vislumbra-se a expansão da base de treinamento para incluir outras geometrias de antenas de microfita (como as circulares) e também a inclusão de um intervalo maior e/ou diferente do utilizado para esse estudo. Além disso, pode-se fazer um estudo mais detalhado, comparando outras técnicas de *Machine Learning* e até mesmo *Deep Learning*, além de uma possível criação de *software* para unificação e facilitação de consulta dos dados.

Referências

1 Toda Matéria. **História e Evolução dos Computadores**. 2026. Disponível em: https://www.todamateria.com.br/historia-e-evolucao-dos-computadores/. Acesso em: 19 ago. 2026.

2 PUCRS. **Inteligência Artificial: o que é e como funciona**. 2026. Disponível em: https://online.pucrs.br/blog/inteligencia-artificial. Acesso em: 19 ago. 2026.

3 SHARMA, K.; PANDEY, G. P. Efficient modelling of compact microstrip antenna using machine learning. **AEU - International Journal of Electronics and Communications**, v. 135, p. 153739, 2021.

4 BALANIS, C. A. **Antenna Theory Analysis and Design**. Hoboken, NJ: Wiley, 2016.

5 Nullspace. **Electromagnetic Simulation Software Comparison: Nullspace EM vs. HFSS vs. CST Studio Suite**. 2026. Disponível em: https://www.nullspaceinc.com/resources/electromagnetic-simulation-software-comparison-nullspace-em-vs-hfss-vs-cst-studio-suite. Acesso em: 15 ago. 2026.

6 RIOS, L. G.; BARBOSA, E. **Engenharia de Antenas**. São Paulo: Edgard Blucher, 2002.

7 DARTORA, C. A. **Ondas Guiadas**. 2015. https://www.eletrica.ufpr.br/cadartora/Documentos/TE053/12-Ondas_Guiadas_e_LT.pdf. Acesso em: 17 out. 2022.

8 BHUNIA, S. **Microstrip Patch Antenna Design, A Novel Approach**. [S.l.]: LAP Lambert Academic Publishing, 2014. ISBN 978-3-659-51925-3.

9 REZENDE, G. P. **Arranjo Linear de Antenas Impressas Faixa Larga Com Diagrama de Irradiação Reconfigurável Em Função Da Frequência Para Aplicações Em 4g E 5g**. Monografia (Bacharelado) — Faculdade de Engenharia Elétrica, Universidade Federal de Uberlândia, Patos de Minas, 2022.

10 MISILMANI, H. M.; NAOUS, T. Machine learning in antenna design: An overview on machine learning concept and algorithms. **2019 International Conference on High Performance Computing & Simulation (HPCS)**, 2019.

11 KDNUGGETS. **Deep Neural Networks**. 2020. https://www.kdnuggets.com/2020/02/deep-neural-networks.html.

12 IBM. **What is supervised learning**. 2018. https://www.ibm.com/topics/supervised-learning. Acesso em: 09 mai. 2023.

13 IBM. **What is unsupervised learning**. 2018. https://www.ibm.com/topics/unsupervised-learning. Acesso em: 09 mai. 2023.

14 SCIKIT-LEARN. **Semi Supervised Learning**. 2010. https://scikit-learn.org/stable/modules/semi_supervised.html. Acesso em: 17 out. 2022.

15 VAPNIK, V. N. Direct methods in statistical learning theory. **The Nature of Statistical Learning Theory**, p. 225–265, 2000.

16 SAXENA, S. **Beginner's Guide to Support Vector Machine(SVM)**. 2021. https://www.analyticsvidhya.com/blog/2021/03/beginners-guide-to-support-vector-machine-svm/. Acesso em: 19 dez. 2023.

17 EVGENIOU, T.; PONTIL, M. Support vector machines: Theory and applications. **Machine Learning and Its Applications**, p. 249–257, 2001.

18 VAPNIK, V. N.; CHERVONENKIS, A. Y. On the uniform convergence of relative frequencies of events to their probabilities. **Theory of Probability & Its Applications**, v. 16, n. 2, p. 264–280, 1971.

19 WAHBA, G. Partial spline models. **Spline Models for Observational Data**, p. 73–94, 1990.

20 GUNN, S. R. Support vector machines for classification and regression. **Support Vector Machines and Their Application in Chemistry and Biotechnology**, p. 15–48, 2011.

21 EDUARDO. **Reconhecimento de Padrões**. 2018. https://www.facom.ufu.br/~backes/pgc204/Aula06-Regressao.pdf. Acesso em: 17 out. 2022.

22 WIKIMEDIA. **Karush–Kuhn–Tucker conditions**. [S.l.]: Wikimedia Foundation, 2023. https://en.wikipedia.org/wiki/KarushâĂŞKuhnâĂŞTucker_conditions. Acesso em: 09 mai. 2023.

23 GOOGLE. **Framing: Key ML Terminology**. [S.l.]: Google, 2022. https://developers.google.com/machine-learning/crash-course/framing/ml-terminology. Acesso em: 17 fev. 2024.

24 BHATTACHARJEE, J. **Some Key Machine Learning Definitions**. [S.l.]: Medium, 2017. https://medium.com/technology-nineleaps/some-key-machine-learning-definitions-b524eb6cb48. Acesso em: 27 fev. 2024.

25 SCIKIT-LEARN. **Pipelines**. [S.l.]: Scikit-Learn, 2024. https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html. Acesso em: 17 fev. 2024.

26 LOUKAS, S. **How and why to Standardize your data: A python tutorial**. [S.l.]: Towards Data Science, 2020. https://towardsdatascience.com/how-and-why-to-standardize-your-data-996926c2c832. Acesso em: 17 fev. 2024.

27 BRANCO, H. **Overfitting e underfitting em Machine Learning**. [S.l.]: Associação Brasileira de Ciência de Dados, 2020. https://abracd.org/overfitting-e-underfitting-em-machine-learning/. Acesso em: 17 fev. 2024.

28 DEVELOPERS, S. learn. **Tuning the hyper-parameters of an estimator**. [S.l.]: Scikit-Learn, 2024. https://scikit-learn.org/stable/modules/grid_search.html. Acesso em: 17 fev. 2024.

29 SCIKIT-LEARN. **Cross-validation: evaluating estimator performance**. [S.l.]: Scikit Learn, 2024. https://scikit-learn.org/stable/modules/cross_validation.html. Acesso em: 17 fev. 2024.

30 BRUCE PETER, A.; GEDECK, P. **Practical statistics for data scientists**. 2. ed. Sebastopol, CA: O'Reilly Media, 2020.