Análise numérica de estruturas eletromagnéticas utilizando *scripts* em Python

Beatriz B. Assis, Elyce M. G. de Lima, Antonio A. F. Junior, Evandro C. Vilas Boas
Laboratório de Cyber Segurança e Internet das Coisas (CS&I Lab.), Instituto Nacional de Telecomunicações - Inatel
beatriz.bastos@get.inatel.br, elyce.maria@get.inatel.br, antonioa@inatel.br, evandro.cesar@inatel.br

***Abstract*—This work presents a qualitative study on numerical analysis of electromagnetic structures using Python scripts. The EMPro electromagnetic analysis tool from Keysight is used with Python scripts to build up a microstrip transmission line three-dimensional model and analyze it. Concepts related to electromagnetism, aspects of the Python programming language, design procedures, such as calculations, selection of numerical methods, and definition of boundary conditions are discussed. The script's numerical results are compared with those obtained from direct user interaction through the human-machine interface. The approaches are equivalent, with easy reproduction of numerical analysis by direct execution of the Python scripts. Meanwhile, direct human-machine interaction requires a longer reproduction time, sensitive to human subjectivity.**

***Index Terms*—Electromagnetism, EMPro, numerical methods, Python, electromagnetic simulation.**

***Resumo*—Este trabalho apresenta um estudo qualitativo sobre análises numéricas de estruturas eletromagnéticas utilizando *scripts* em linguagem Python. Emprega-se a ferramenta de análise eletromagnética EMPro da empresa Keysight e aplicam-se *scripts* em Python para a construção do modelo tridimensional e análise de uma linha de transmissão impressa. Abordam-se conceitos relacionados ao eletromagnetismo, aspectos da linguagem de programação Python, procedimentos de projeto, tais como cálculos, seleção de métodos numéricos e definição de condições de contorno. Comparam-se os resultados numéricos obtidos por meio de *scripts* com aqueles oriundos da interação direta do usuário pela interface homem-máquina. Verificam-se equivalência entre as abordagens e a fácil reprodução da análise numérica pela execução direta dos *scripts* em Python. Enquanto, a interação direta homem-máquina exige maior tempo de reprodução, sendo susceptível a subjetividade humana.**

***Palavras chave*—Eletromagnetismo, EMPro, métodos numéricos, Python, simulação eletromagnética.**

I. Introdução

A simulação computacional tornou-se uma ferramenta importante para o desenvolvimento de produtos no âmbito da Engenharia. A Engenharia assistida por simulação ou Engenharia assistida por computador (CAE, *Computer Aided Engineering*) fornece meios computacionais que possibilitam a análise e exploração de soluções distintas para um determinado problema, reduzindo custos durante o desenvolvimento de um projeto. Quando comparada ao uso de CAD e análise convencional e testes de protótipos, a simulação computacional concentra as alterações de um projeto nas fases iniciais, reduzindo a construção sequencial de protótipos ou provas de conceito. Essa ferramenta proporciona maior assertividade na fase de prototipagem e validação mediante processos de otimização. Dessa forma, obtêm-se baixo custo de produção e maior rentabilidade.

Dentre as diversas áreas, a simulação computacional possui ampla aplicação em projetos relacionados ao eletromagnetismo [1, 2, 3]. Desenvolvem-se estruturas eletromagnéticas como guias de ondas, antenas e filtros por meio de simulações eletromagnéticas [4]. Os programas de simulação eletromagnética fornecem ferramentas para a construção de modelos numéricos que se assemelham ao produto final, permitindo aplicar diferentes materiais para otimização de resultados. Assim como, utilizam-se processos de otimização para obter resultados ótimos em conformidade com as técnicas de fabricação.

Estruturam-se os programas de simulação em modelagem matemática, implementação numérica ou cálculo de solução, ambiente gráfico e interface homem-máquina. A modelagem matemática compreende o conjunto de equações e fórmulas previamente definidas pelo homem e que modelam os fenômenos naturais, como os eletromagnéticos. O cálculo de solução emprega métodos numéricos que permitem resolver conjuntos de equações e/ou fórmulas para obtenção das soluções, que seriam impossíveis por meios analíticos. O ambiente gráfico proporciona a visualização científica por meio do tratamento das soluções e exibição de resultados. A interface homem-máquina integra as partes supracitadas, tornando-as de fácil acesso ao usuário final.

Dentre os programas de simulação eletromagnética, encontra-se o PathWave EM Design, referenciado como EMPro [5]. Com o aprimoramento dessa ferramenta ao longo dos anos, tornou-se possível empregar a linguagem de programação em Python para a escrita de projetos eletromagnéticos diretamente na ferramenta [6, 7]. Contudo, essa área é pouco difundida entre os profissionais que atuam no desenvolvimento de estruturas eletromagnéticas. Portanto, este trabalho tem por objetivo apresentar um estudo qualitativo sobre análises numéricas de estruturas eletromagnéticas utilizando *scripts* em linguagem Python. Emprega-se a ferramenta de análise eletromagnética EMPro e aplicam-se *scripts* em Python para a construção do modelo tridimensional e análise de uma linha de transmissão impressa. Estrutura-se o trabalho em quatro seções. Na Seção II, discutem-se os princípios e conceitos básicos relacionados a execução de uma simulação eletromagnética. Exploram-se *scripts* em Python na Seção III. Aborda-se a construção do modelo numérico e a simulação de uma linha de transmissão impressa por *scripts*, onde comparam-se os resultados com aqueles obtidos por meio de interação direta do usuário com a interface do EMPro. Conclusões e comentários finais encontram-se na Seção IV.

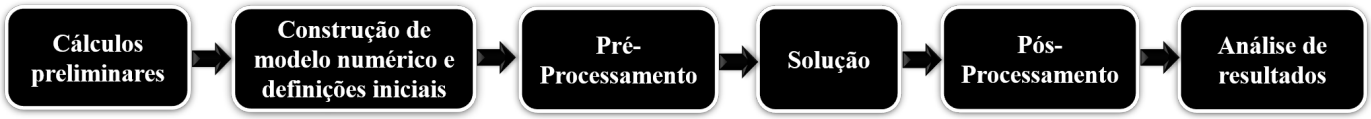

Fig. 1. *Etapas de um processo de simulação.*

II. Princípios básicos em simulação eletromagnética

Um processo de simulação envolve no mínimo seis etapas, conforme estruturado na Fig. 1. Inicialmente, têm-se os cálculos ou decisões preliminares que resultam em uma solução ou modelo inicial. Essa etapa não possui relação direta com os programas de simulação e fundamenta-se nos conhecimentos adquiridos pelo profissional ao longo de sua formação. Com a abordagem inicial, constrói-se o modelo numérico inicial no programa. Atribuem-se materiais para as geometrias e definem-se o método numérico, domínio computacional e condições de contorno. Discretizam-se as geometrias para obtenção da malha inicial, empregando-a para a solução e obtenção de resultados. Em alguns casos, o pré-processamento inclui a adaptação de malha para que ocorra a convergência. Por fim, pós processa-se a solução e extrai-se os resultados, que devem ser interpretados pelo projetista.

As equações de Maxwell correspondem a um sistema de equações diferenciais acopladas, tendo duas funções desconhecidas: os campos elétrico E e magnético H. Definem-se esses campos dentro de um determinado espaço de solução ou domínio computacional e sob condições de contorno, estabelecidas de acordo com o problema em análise. Geralmente, problemas de natureza eletromagnética também envolvem uma fonte de irradiação primária e independente dos campos E e H, do material e do meio. As equações de Maxwell apresentam soluções analíticas apenas para caso simples como, por exemplo, um elemento de corrente infinitesimal. Referenciam-se esses casos como problemas de soluções canônicas. Contudo, grande parte das aplicações reais em eletromagnetismo não se enquadram nessa categoria. Logo, faz-se necessária uma abordagem numérica para solução das equações de Maxwell, resultando no uso de métodos numéricos.

A adequação do método numérico ao problema em análise depende de um conjunto de fatores. Destacam-se limitações da abordagem numérica e a natureza do modelo numérico em relação à aplicação. Os diversos métodos numéricos se diferem quanto a complexidade, demanda por recursos computacionais e domínio de solução. Por exemplo, a densidade de discretização de um método reflete diretamente na precisão da solução do problema, definindo o erro de discretização. Dependendo da aplicação, tem-se maior ou menor tolerância a esse erro, impactando na escolha do método. Por outro lado, a natureza do modelo numérico refere-se ao comportamento previamente determinado pelo problema real e permite definir entre o uso de técnicas numéricas de solução no domínio do tempo ou da frequência. No campo do eletromagnetismo, destacam-se o uso do método de domínio do tempo com diferenças finitas (FDTD, *Finite-difference time-domain*), método dos elementos finitos (FEM, *Finite Element Method*) e o método dos momentos (MoM, *Method of Moments*) [1, 2, 3].

A. Métodos numéricos

O FDTD emprega a discretização das equações de Maxwell no domínio do tempo para solucioná-las diretamente. Já o FEM e o MoM resolvem as equações de Maxwell indiretamente por meio de suas formas integrais, resultando em uma solução intermediária para determinar as componentes de campo elétrico e magnético. Independentemente do método, a abordagem numérica utiliza a discretização dos campos dentro do domínio computacional como princípio de solução. Dessa forma, obtém-se uma distribuição discreta dentro de uma grade (ou malha) de pontos finitos e calcula-se uma solução para os campos em um determinado ponto discreto desse espaço. O FDTD discretiza uma estrutura dividindo-a em elementos uniformes para a formação da malha. Esses elementos possuem formato quadrado ou retangular para geometrias bidimensionais e formato cúbico para geometrias tridimensionais [1, 2].

O FEM se baseia na solução volumétrica, em que seccionase o domínio de solução em pequenos elementos triangulares ou tetraedros, referenciados como elementos finitos [1]. Essa abordagem garante flexibilidade e fácil representação de estruturas complexas e domínio de solução durante o processo de discretização. Além disso, o FEM permite a criação de uma malha adaptativa de acordo com as características físicas do problema. Criam-se malhas densas em regiões onde a variação dos campos é maior para obter-se uma solução mais precisa. Em regiões da geometria em que essa variação é menor, aplicam-se malhas com elementos finitos maiores. Essa característica colabora para otimização no uso de recursos computacionais.

O MoM divide a estrutura em diversos segmentos lineares e planos, cujas dimensões são inferiores ao comprimento de onda de interesse [1, 2, 3]. Em seguida, utiliza-se uma função expandida em uma combinação linear de N funções para representar a corrente elétrica desconhecida. Aplicam-se operações matriciais para gerar a equação matricial, solucionando-a para a distribuição de corrente na superfície da estrutura. Emprega-se essa distribuição de corrente para determinar o campo elétrico em um ponto do espaço por meio da contribuição de cada elemento.

B. PathWave EM Design – EMPro e scripts em Python

PathWave EM Design, referenciado como EMPro, é uma ferramenta de simulação eletromagnética 3D da empresa Keysigth Technologies [1]. Essa ferramenta permite a análise de desempenho eletromagnético de estruturas 3D em radiofrequência (RF) e micro-ondas. O EMPro possui um ambiente para modelagem de estruturas 3D com a possibilidade de importação/exportação de arquivos CAD. Pode-se realizar as

análises eletromagnéticas por meio do método numérico FEM ou FDTD. Dentre as aplicações do EMPro, citam-se a análise de circuitos integrados, módulos RF com múltiplas camadas, componentes de RF (ex.: conectores), placas de circuito impresso, conectores de alta-velocidade (ex.: HDMI) e antenas. Além disso, emprega-se a ferramenta em análises no campo aeroespacial, de compatibilidade eletromagnética (EMC, *electromagnetic compatibility*) e/ou interferência eletromagnética (EMI, *electromagnetic interference*).

O EMPro possui em sua interface homem-máquina um campo voltado para a programação em linguagem Python, viabilizando a construção de *scripts* para análise de estruturas eletromagnéticas [**7**]. Existem alguns grupos de comandos necessários para à implementação de *scrips*. Esses comandos encontra-se agrupados em uma biblioteca nativa do programa, a *empro*. Destacam-se comandos usuais como *clear*, que permite limpar qualquer comando escrito anteriormente, assim como geometrias e excitações; *activeProject* para iniciar um novo projeto e validar algumas ações; e *name* para renomear geometrias. Exploram-se esses e outros comandos na Seção III.

III. Projeto e Simulação de uma Linha de Transmissão Impressa em Script Python

Nessa seção, avalia-se qualitativamente a análise de uma estrutura eletromagnética em relação ao uso de *scripts* em Python e interface homem-máquina. Para o estudo, considera-se o projeto de uma linha de transmissão impressa com impedância característica (Z_o) de 50 Ω construída sobre laminado dielétrico FR4, cuja espessura (h) e constante dielétrica (ε_r) são respectivamente iguais a 1,6 mm e 4,6. Determina-se a largura da microlinha de fita (w) por meio de [4]:

$$\frac{w}{h} = \frac{e^u}{8} + \frac{1}{4e^u} \tag{1}$$

sendo u:

$$u = \frac{Zo\sqrt{2(\varepsilon_r+1)}}{120} + \frac{1}{2}\left(\frac{\varepsilon_r - 1}{\varepsilon_r + 1}\right)\left[\ln\left(\frac{\pi}{2}\right) + \frac{1}{\varepsilon_r}\ln\left(\frac{\pi}{4}\right)\right] \tag{2}$$

Para o projeto, obteve-se w = 2,96 mm. Definiu-se as dimensões do substrato e plano terra em: 40 mm x 20 mm. Aplicaram-se essas dimensões na programação dos *scripts* por meio do ambiente de programação em Python do EMPro denominado de *Scripting* [7]. Para iniciar um projeto no EMPro, empregam-se comandos específicos da biblioteca *empro*, conforme indicado a seguir. Posteriormente, iniciam-se as etapas discutidas na Seção II, conforme visto na Fig 1.

```
#Criando um novo projeto:
empro . activeProject . clear ()
```

A. Construção da geometria e atribuição de materiais

Para construir geometrias tridimensionais diversas por meio de *scripts*, deve-se criar uma lista de geometrias, como visto na linha 3 do bloco de códigos a seguir. Posteriormente, emprega-se o comando *empro.geometry* e especifica-se a geometria e suas respectivas dimensões. Utilizam-se comandos específicos como *box* para paralelepípedos, *sphere* para esferas, *torus* para toroides, *prism* para prisma, *pyramid* para pirâmides, *cylinder* para cilindro, *cone* para cone e *helix* para hélices. Após criar a geometria, adiciona-a à lista de geometria do projeto, como demonstra a linha 6. Por conseguinte, atribui-se um material a geometria, cuja inclusão deve ser feita previamente ao projeto, linha 10. Para o exemplo, adicionou-se o material FR4 presente na biblioteca do programa EMPro. Por fim, pode-se renomear a geometria para fins de identificação.

```
#Criando o substrato:
model = empro . geometry . Model ()
model . recipe . append ( empro . geometry . Box ("
    40 mm", "1.6 mm", "20 mm"))
empro . activeProject . geometry . append ( model )

#Adicionando o material FR4 ao projeto:
empro . activeProject . materials . append (empro .
    toolkit . defaultMaterial ("FR-4"))

#Atribuindo o material FR4 ao substrato:
empro . activeProject . geometry [0]. material =
    empro . activeProject . materials ["FR-4"]

#Renomeando a geometria do substrato:
parts = empro . activeProject . geometry
parts [0]. name = " Substrato "
```

Para os laminados de cobre que representam a linha de transmissão impressa e o plano de terra, utilizam-se geometrias planares. A construção desse tipo de geometria envolve operações com seus respectivos vértices, que formam parâmetros de entradas para comandos específicos da biblioteca *empro*. Utilizam-se algumas funções que podem ser vistas no Apêndice A desse documento. No código a seguir, demostra-se como definir os vértices dessas geometrias, adicionando-as à lista de geometrias do projeto. Assim como, atribui-se o material cobre à esses laminados.

```
#Definindo as dimensoes dos vertices do plano terra:
verts = [(" -20 mm", " -10 mm"),("20 mm", " -10 mm")
    ,("20 mm", " 10 mm"),(" -20 mm", "10 mm")]

#Criando a geometria do plano terra:
wirebody = makePolygon ( verts )
sheet = sheetFromWireBody ( wirebody )
sheet . name = " Ground "

#Adicionando a geometria ao projeto:
parts . append ( sheet )

#Adicionando o material cobre ao projeto e
    atribuindo ao plano terra:
mats = empro . activeProject . materials
mats . append ( empro . toolkit . defaultMaterial ("
    Cu") )
parts [ -1]. material = mats ["Cu"]

#Definindo as dimensoes dos vertices da linha de
    transmissao:
verts = [(" -20 mm", "0 mm"),("20 mm", "0 mm")]
wirebody = makePolyLine ( verts )

#Parametrizando a espessura da linha e atribuindo o
    valor calculado:
empro . activeProject . parameters . append ("
    myWidth", "2.96 mm")
model = empro . geometry . Model ()

#Atribuindo as dimensoes a linha de transmissao:
trace = empro . geometry . Trace ( wirebody )
trace . width = " myWidth "
model . recipe . append ( trace )
model . name = " Microstrip "
```

```
parts . append ( model )

#Posicionando a linha de transmissao no topo do substrato:
parts [ -1]. coordinateSystem . anchorPoint = (0, 0, "1.6 mm")

#Adicionando o material cobre ao projeto e atribuindo a linha de transmissao
mats = empro . activeProject . materials
mats . append ( empro . toolkit . defaultMaterial ("Cu") )
parts [ -1]. material = mats ["Cu"]
```

A execução do código no Apêndice até a linha 96, permite a construção do modelo numérico tridimensional da linha de transmissão, como visto na Fig 2. Posicionaram-se as geometrias em torno da origem dos eixos de forma simétrica.

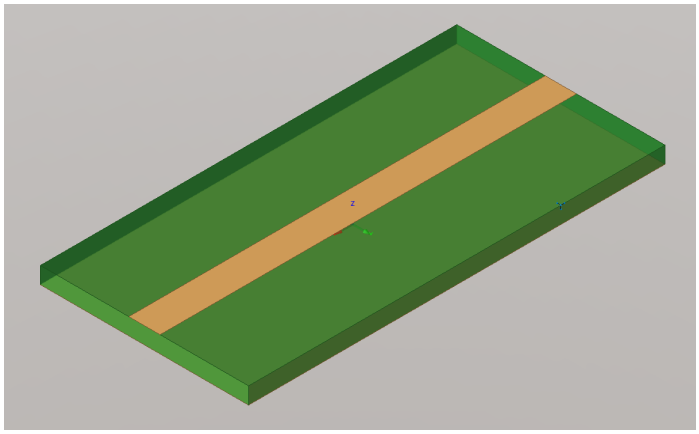

Fig. 2. *Geometria da linha de transmissão.*

B. Atribuição de excitação

Para reproduzir a excitação em um modelo numérico, utilizam-se de estruturas definidas como excitações. O EM-Pro possui três tipos: *Line Port*, *Sheet Port* e *Wave Port*. Cada excitação possui características peculiares relacionadas a construção e aplicação. Essa discussão não está no escopo desse trabalho e recomenda-se que o leitor consulte [5] para detalhes. No projeto, definiu-se a excitação *Line Port* formada por um vetor, cujas extremidades indicam a diferença de potência entre as geometrias. Posicionaram-se duas excitações nas extremidades da linha de transmissão impressa da Fig 2. No bloco abaixo, definem-se os principais comandos para acrescentar as excitações ao longo da geometria. Em seguida, definem-se a tensão e a impedância dessas fontes de excitação.

```
#Adicionando excitacoes:

#Line port 1
feed = empro . components . CircuitComponent ()
feed . name = "My Feed " #nomeando a porta
feed . definition = empro . activeProject . defaultFeed ()
feed . tail = ("-20 mm", 0 , 0) #definindo o inicio do vetor
feed . head = ("-20 mm", 0 , "+1.6 mm") #definindo o inicio do vetor
empro . activeProject . circuitComponents . append ( feed )

#Line port 2
feed = empro . components . CircuitComponent ()
feed . name = "My Feed " #nomeando a porta
feed . definition = empro . activeProject . defaultFeed ()
feed . tail = ("20 mm", 0 , 0) #definindo o inicio do vetor
feed . head = ("20 mm", 0 , "+1.6 mm") #definindo o inicio do vetor
empro . activeProject . circuitComponents . append ( feed )

#Definindo a tensao e impedancia da fonte de excitacao:
feedDef = empro . components . Feed ("My Voltage Source")
feedDef . feedType = "Voltage"
feedDef . amplitudeMultiplier = "1 V"
feedDef . impedance . resistance = "50 ohm"
feed . definition = feedDef
```

C. Definição de método numérico e execução da simulação

Após construir a geometria do modelo, atribuir materiais e excitações, tem-se um modelo numérico apto a execução da análise numérica. Logo, deve-se incluir os comandos relacionados à definição do método numérico e seus parâmetros de análise. Para este trabalho, selecinou-se o FEM. No bloco de códigos abaixo, definem-se os parâmetros necessários para a execução da análise numérica pelo FEM: tipo de análise (adaptativa), frequências mínima e máxima da faixa de análise (1 à 10 GHz), quantidade de pontos de discretização (101), abordagem da solução (Direta) e operação de malha (auto adaptativa). Por fim, inicia-se a simulação. No código apresentado no Apêndice, acrescentaram-se linhas de códigos que permitem identificar e avisar ao usuário quando a simulação numérica encerrou.

```
#Iniciando a simulacao e configurando parametros
simSetup = empro . activeProject . simulationSettings

simSetup . engine = "FemEngine" #selecionando o metodo FEM

freqPlans = simSetup . femFrequencyPlanList ()
freqPlans . clear ()

plan = empro . simulation . FrequencyPlan ()
plan . type = "Adaptive" #tipo de analise
plan . startFrequency = " minFreq " #frequencia minima de analise
plan . stopFrequency = " maxFreq " #frequencia maxima de analise
plan . samplePointsLimit = 101 #numero de pontos
freqPlans . append ( plan )

params = empro . activeProject . parameters
params . setFormula (" minFreq ", "1 GHz ")
params . setFormula (" maxFreq ", "10 GHz ")

#Definindo o solver:
simSetup . femMatrixSolver . solverType = "MatrixSolverDirect"
simSetup . femMeshSettings . autoConductorMeshing = True
```

D. Comparação de resultados

Pode-se obter os resultados diretamente do programa por meio de comandos específicos e uma API (*Application Programming Interface*). Nesse caso, utilizou-se a interface homem máquina. Na Fig 3, apresentam-se os parâmetros *S* para

a linha de transmissão impressa para um modelo construído por meio da interface de usuário em um modelo construído e simulado por meio de comandos em Python, agrupados em *script*. Verifica-se a concordância numérica entre os resultados, validando a abordagem proposta. Destaca-se que o uso de *scripts* em Python permite automatizar o processo de construção e configuração de simulações eletromagnéticas, reduzindo tempo de projetistas na configuração de passos comuns à diversos projetos.

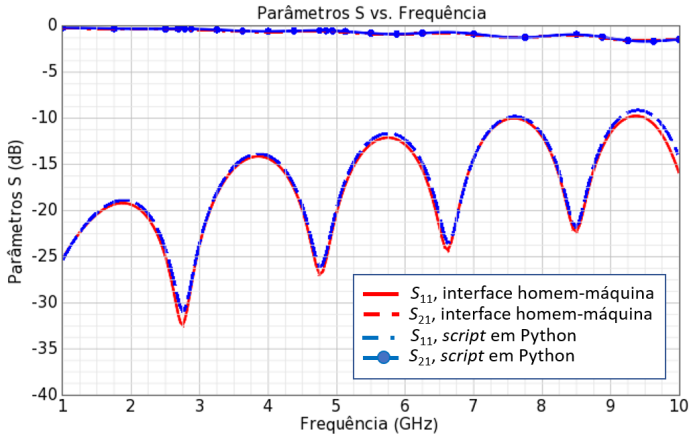

Fig. 3. *Parâmetros S para a linha de transmissão impressa.*

IV. Conclusão

Esse trabalho apresentou um estudo qualitativo sobre análises numéricas de estruturas eletromagnéticas utilizando *scripts* em linguagem Python. Para o estudo, empregou-se a ferramento EMPro da Keysight e desenvolveram-se *scripts* para viabilizar a análise numérica de uma linha de transmissão impressa. Comparam-se os parâmetros *S* do modelo obtido por meio de *scripts* com aqueles oriundos de um segundo modelo analisado pela interação direta do usuário com a interface da ferramenta. Obteve-se concordância entre as curvas, validando a abordagem e sua adoção na automação total ou parcial de um projeto. Etapas repetitivas como configuração de parâmetros de simulação configuram um exemplo de automação por meio de *scripts* em Python, que podem ser executados após a conclusão das etapas que precedem-na.

Agradecimentos

Os autores agradecem ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio na execução das atividades propostas, ao Instituto Nacional de Telecomunicações (Inatel) e ao Laboratório de Segurança Cibernética e Internet das Coisas (CS&I Lab.) por todo o suporte técnico oferecido.

Referências

[1] J. L. Volakis, A. Chatterjee e L. C. Kempel. *Finite Element Method for Electromagnetics*. New York: IEEE Series, 1998.

[2] A. F. Peterson, S. L. Ray e R. Mittra. *Computational Methods for Electromagnetics*. New York: IEEE Series, 1998, 1998.

[3] M. N. O. Sadiku. *Numerical Techniques in Electromagnetics*. 2ª ed. New York: CRC Press, 2001.

[4] J. A. J. Ribeiro. *Engenharia de Antenas: Fundamentos, Projetos e Aplicações*. 1ª ed. São Paulo: Editora Érica, 2012.

[5] Keysight. *PathWave EM Design (EMPro*. URL: https://www.keysight.com/br/pt/products/software/pathwave-design-software/pathwave-em-design-software.html (acesso em 03/01/2021).

[6] M. Lie Hetland. *Python Algorithms Mastering Basic Algorithms in the Python Language*. 1ª ed. Apress, 2010.

[7] Keysigth. *Keysight EMPro Scripting Cookbook*. 5ª ed. 2020.

V. Apêndice

A. Código final para a construção e simulação de uma linha de transmissão impressa

```
#Codigo para simulacao de uma linha de transmissao impressa:

def makePolyLine ( vertices , sketch =None , name = None ):
  """
    Funcao para conectar os pontos do poligono:
    - vertices: sequencia de coordenadas (x,y,z) a serem conectadas.
    - sketch[opcional]: se fornecido, acrescenta uma polyline a ele. Caso contrario, um novo Sketch sera criado.
    - name[opcional]: nome do scketch.
    """
  from empro import geometry
  sketch = sketch or geometry . Sketch ()

  #Enquanto o inicio e o fim da geometria estao dentro dos vertices,adicionam as linhas do esboco
  if name :
    sketch . name = name
  for tail , head in zip ( vertices [: -1] , vertices [1:]) :
    sketch . add ( geometry . Line (tail , head ))
  return sketch

def makePolygon ( vertices , sketch =None , name = None ):
  """
    Funcao para criar o poligono.
    A Polilinha retorna o vertice das extremidades do poligono,seu esboco e nome
  """
  return makePolyLine ( vertices + vertices [:1] , sketch = sketch , name = name )

def sheetFromWireBody ( wirebody , name = None ):
  """
    Funcao para criar Sheet Body cobrindo a Wire Body.
    Um novo modelo  retornado.
    Wirebody e clonado para que o original nao seja danificado .
  """
  from empro import geometry
  model = geometry . Model ()
  #Na receita da geometria cria um clone da wirebody , pois eh necessario  ter nas duas tampas
  model . recipe . append ( geometry . Cover ( wirebody . clone ()))
  model . name = name or wirebody . name
  return model
```

```

#Criando um novo projeto:
empro . activeProject . clear ()

#Criando o substrato:
model = empro . geometry . Model ()
model . recipe . append ( empro . geometry . Box ("40 mm", "1.6 mm", "20 mm"))
empro . activeProject . geometry . append ( model )

#Adicionando o material FR4 ao projeto:
empro . activeProject . materials . append (empro . toolkit . defaultMaterial ("FR-4"))

#Atribuindo o material FR4 ao substrato:
empro . activeProject . geometry [0]. material = empro . activeProject . materials ["FR-4"]

#Renomeando a geometria do substrato:
parts = empro . activeProject . geometry
parts [0]. name = " Substrate "

#Definindo as dimensoes dos vertices do substrato:
verts = [(" -20 mm", " -10 mm"),("20 mm", " -10 mm"),("20 mm", " 10 mm"),(" -20 mm", "10 mm")]

#Criando a geometria do plano terra:
wirebody = makePolygon ( verts )
sheet = sheetFromWireBody ( wirebody )
sheet . name = " Ground "

#Adicionando a geometria ao projeto:
parts . append ( sheet )

#Adicionando o material cobre ao projeto e atribuindo ao plano terra:
mats = empro . activeProject . materials
mats . append ( empro . toolkit . defaultMaterial ("Cu") )
parts [ -1]. material = mats ["Cu"]

#Definindo as dimensoes dos vertices da linha de transmissao:
verts = [(" -20 mm", "0 mm"),("20 mm", "0 mm")]
wirebody = makePolyLine ( verts )

#Parametrizando a espessura da linha e atribuindo o valor calculado:
empro . activeProject . parameters . append ("myWidth", "2.96 mm")
model = empro . geometry . Model ()

#Atribuindo as dimenses ao substrato:
trace = empro . geometry . Trace ( wirebody )
trace . width = " myWidth "
model . recipe . append ( trace )
model . name = " Microstrip "
parts . append ( model )

#Posicionando a linha de transmissao no topo do substrato:
parts [ -1]. coordinateSystem . anchorPoint = (0, 0, "1.6 mm")

#Adicionando o material cobre ao projeto e atribuindo a linha de transmissao
mats = empro . activeProject . materials
mats . append ( empro . toolkit . defaultMaterial ("Cu") )
parts [ -1]. material = mats ["Cu"]

#Adicionando excitacoes:

#Line port 1
feed = empro . components . CircuitComponent ()
feed . name = "My Feed " #nomeando a porta
feed . definition = empro . activeProject . defaultFeed ()
feed . tail = ("-20 mm", 0 , 0) #definindo o inicio do vetor
feed . head = ("-20 mm", 0 , "+1.6 mm") #definindo o inicio do vetor
empro . activeProject . circuitComponents . append ( feed )

#Line port 2
feed = empro . components . CircuitComponent ()
feed . name = "My Feed " #nomeando a porta
feed . definition = empro . activeProject . defaultFeed ()
feed . tail = ("20 mm", 0 , 0) #definindo o inicio do vetor
feed . head = ("20 mm", 0 , "+1.6 mm") #definindo o inicio do vetor
empro . activeProject . circuitComponents . append ( feed )

#Definindo a tensao e impedancia da fonte de excitacao:
feedDef = empro . components . Feed ("My Voltage Source")
feedDef . feedType = "Voltage"
feedDef . amplitudeMultiplier = "1 V"
feedDef . impedance . resistance = "50 ohm"
feed . definition = feedDef

#Iniciando a simulacao e configurando parametros
simSetup = empro . activeProject . simulationSettings

simSetup . engine = "FemEngine" #selecionando o metodo FEM

freqPlans = simSetup . femFrequencyPlanList ()
freqPlans . clear ()

plan = empro . simulation . FrequencyPlan ()
plan . type = "Adaptive" #tip de analise
plan . startFrequency = " minFreq " #frequencia minima de analise
plan . stopFrequency = " maxFreq " #frequencia maxima de analise
plan . samplePointsLimit = 101 #numero de pontos
freqPlans . append ( plan )

params = empro . activeProject . parameters
params . setFormula (" minFreq ", "1 GHz ")
params . setFormula (" maxFreq ", "10 GHz ")

#Definindo o solver:
simSetup . femMatrixSolver . solverType = "MatrixSolverDirect"
simSetup . femMeshSettings . autoConductorMeshing = True

#Salvando o projeto:
empro . activeProject . saveActiveProjectTo (r"C:\tmp\ linesimulation .ep")

#Iniciando a simualacao
sim = empro . activeProject . createSimulation ( True )

#Comandos para indicar o fim da simulacao:
from empro . toolkit . simulation import wait
empro .gui . activeProjectView (). showScriptingWindow ()
print (" waiting ... ")
wait ( sim )
print (" Done !")
print (sim . status )
```