DOI: 10.48011/sbse.v1i1.2333

Ferramentas de Prototipação Aplicadas a Sistemas de Potência: Matlab versus Python

Luciano O. Daniel* **Sergio L. Varricchio****

**Universidade Federal Fluminense, Niterói, CEP 24240-210*
BRASIL (e-mail: luciano.oliveira.daniel@gmail.com)
***Centro de Pesquisas de Energia Elétrica, Rio de Janeiro, CEP 21941-911*
BRASIL (e-mail: slv@cepel.br)

Abstract: The main objective of this work is to present a comparison between Matlab and Python tools applied to the creation of prototypes for the analysis of electrical power systems. Comparative precision tests will be performed between the tools for the main time-frequency transformation methods, numerical quadrature and numerical linear algebra applied to the analysis of power systems. The authors' point of view will also be presented regarding the facilities offered by these tools, such as programming environments, source code text editors, graphical functions and error debuggers.

Resumo: O principal objetivo deste trabalho é apresentar uma comparação entre as ferramentas Matlab e Python aplicadas à criação de protótipos para análise de sistemas elétricos de potência. Serão realizados testes comparativos de precisão entre as ferramentas para os principais métodos de transformação tempo-frequência, de quadratura numérica e de álgebra linear numérica aplicados à análise de sistemas de potência. Também será apresentada a visão crítica dos autores em relação às facilidades oferecidas por tais ferramentas, como ambientes de programação, editores de texto do código-fonte, funções para traçado de gráficos e depuradores de erros.

Keywords: Matlab; Python; Prototyping; Power Systems; Numerical Computation.

Palavras-chaves: Matlab; Python; Prototipação; Sistemas de Potência; Computação Numérica.

1. INTRODUÇÃO

Nos centros de pesquisas e universidades ao redor do mundo, existe a necessidade da constante utilização de ferramentas/ambientes computacionais que facilitem e acelerem os testes e o desenvolvimento inicial (protótipo) de métodos, modelos matemáticos, algoritmos e provas de conceito aplicados a sistemas elétricos de potência e equipamentos associados.

O programa Matlab é uma ferramenta consagrada para este fim, que possui um ambiente de programação com diversos métodos matemáticos integrados e otimizados (*toolboxes*) e facilidades para a visualização de resultados (e.g. gráficos e tabelas) e integração com planilhas eletrônicas (e.g. Excel). De particular importância dentre estes métodos, para a análise de redes elétricas, são os de transformação tempo-frequência (e.g. FFT - *Fast Fourier Transform*), de quadratura numérica (e.g. Gauss-Legendre) e de álgebra linear numérica (e.g. fatorações LU, QR e QZ, autovalores e autovetores, valores singulares e mínimos quadrados). Outro recurso matemático disponível e de grande importância, por exemplo, para o cálculo de matrizes formadas por derivadas parciais (e.g. Jacobianas e Hessianas) é o cálculo simbólico envolvendo funções analíticas. Além destes, o SIMULINK é outro recurso importante para simulação rápida de circuitos elétricos, sistemas de controle, sistemas contendo equipamentos de eletrônica de potência, dentre outros.

Apesar das vantagens citadas, o MATLAB apresenta um custo de licença de uso relativamente alto e que aumenta na medida em que se incluam mais *toolboxes* no pacote licenciado.

Por outro lado, existem outras linguagens e ambientes de programação disponíveis atualmente que podem ser utilizadas para os mesmos fins de prototipação para sistemas de potência, como: Python, Julia, Octave, Scilab e Mathematica.

As quatro primeiras ferramentas são gratuitas e de código fonte aberto (*open source*), enquanto a última é paga, a exemplo do Matlab. Entre estas, a linguagem que vem apresentando o maior crescimento de sua comunidade de usuários é o Python. Um dos principais motivos para isto é a grande quantidade de métodos matemáticos integrados (*Python Packages*) disponíveis. Outra característica importante do Python é sua rápida curva de aprendizagem, o que facilita bastante sua eventual adoção em substituição à uma determinada linguagem de prototipação.

Desta forma, o principal objetivo deste trabalho é apresentar comparações entre Matlab e Python, objetivando analisar suas precisões em cálculos comumente utilizados em protótipos para sistemas elétricos de potência, como aqueles já citados.

Também será apresentada a visão crítica dos autores em relação às facilidades oferecidas pelas ferramentas, como ambientes de programação, editores de texto do código-fonte, funções para traçado de gráficos (histogramas, nuvem de pontos, curvas, etc.) e depuradores de erros (*debug*).

Adicionalmente, será avaliada a possibilidade de se estender o uso da linguagem Python, não apenas como ferramenta de prototipação, mas também para desenvolvimento de versões comerciais. Neste contexto, será explorada a capacidade de compilação (transformação do código fonte em linguagem de máquina).

2. MÉTODOS NUMÉRICOS APLICADOS A SISTEMAS DE POTÊNCIA

A seguir serão apresentados, de maneira resumida, os principais cálculos aplicados a sistemas elétricos de potência em abordagens numéricas computacionais.

2.1 Transformações Tempo-Frequência

2.1.1 Transformada de Laplace

A transformada de Laplace (Poularikas and Seely 2010) é definida por uma integral imprópria que permite reduzir a complexidade do processo de análise do comportamento de um sistema de potência, pois converte equações diferenciais em equações algébricas e convolução de funções em meros produtos das mesmas. Este operador linear é amplamente utilizado na teoria de controle de sistemas de potência quando representados por funções de transferência (Ogata 2010). Esta transformada também é utilizada em análises de transitórios eletromagnéticos em sistemas de potência (Moreno and Ramirez 2008), (Gómez and Uribe 2009).

2.1.2 Transformada Discreta de Fourier

A transformada de Fourier de tempo discreto (DTFT) aplica-se a funções discretas (Oppenheim and Schafer 1998). Quando tal função é periódica, tem-se um caso particular que é a Transformada Discreta de Fourier (DFT). Um dos algoritmos mais eficientes para o cálculo computacional da DFT (Cooley and Tukey 1965) é conhecido como Transformada Rápida de Fourier (FFT - *Fast Fourier Transform*).

Embora a análise de Fourier seja muito utilizada em processamento de sinais, existem análises de sistemas de potência que a utilizam como, por exemplo, a avaliação de harmônicos (Mayoral et al. 2017), (Yong and Bolin 2009), (Rehman et al. 2015), (Liu et al. 2011) e de transitórios eletromagnéticos (Segundo-Ramirez et al. 2020).

2.2 Quadratura Numérica

A quadratura (ou integração) numérica aproxima, dentro de um intervalo, a integral de funções reais ou complexas. Existem vários métodos possíveis para o cálculo da quadratura de uma função como, por exemplo, Newton-Cotes, Gauss-Legendre, Gauss-Kronrod, Gauss-Tchebycheff, Romberg, dentre outras (Ames and Brezinski 1993), (Kythe and Schäferkotter 2004). Como exemplo de aplicação de quadratura numérica a sistemas de potência, pode-se citar o cálculo de resíduos associados a polos dominantes de funções de transferência (FTs) (Varricchio, Freitas, Martins, et al. 2015). Os polos dominantes e resíduos associados de uma FT constituem o seu modelo racional (MR) de ordem reduzida, podendo ser utilizado para a construção de equivalentes de alta fidelidade para estudos de transitórios eletromagnéticos (Campello et al. 2020).

2.3 Álgebra Linear Numérica

2.3.1 Autovalores e Autovetores

Um escalar $\lambda \in \mathbb{C}$ é um autovalor de uma matriz $\mathbf{A} \in \mathbb{R}^{n \times n}$ se existe um vetor $\mathbf{x} \in \mathbb{C}^n$ não nulo tal que $\mathbf{Ax} = \lambda\mathbf{x}$. Os vetores $\mathbf{x}$ que satisfazem a igualdade anterior são chamados de autovetores (à direita) de $\mathbf{A}$ associados ao autovalor λ. Outro ponto importante é que a matriz $\mathbf{A}$ é invertível (ou não-singular) se $\det(\mathbf{A}) \neq \mathbf{0}$. Na modelagem dos sistemas de controle, o cálculo dos autovalores é fundamental tanto na avaliação da estabilidade do sistema de potência estudado quanto no projeto de compensadores ou controladores (Ogata 2010) através, por exemplo, dos diagramas de lugar-das-raízes e das técnicas de realocação de polos para melhora do desempenho dinâmico do sistema.

Os autovetores são úteis, por exemplo, na obtenção de matrizes de mudança de base que determinam as formas controláveis e observáveis do sistema modelado em espaço de estados (Kundur 2017). Os autovetores também tem uma relação direta de ponderação das parcelas das respostas no tempo. Na modelagem da dinâmica de sistemas de potência por espaço de estados ou por sistemas descritores (Daniel et al. 2013), (Daniel and Gomes Jr 2017), os autovalores estão associados aos polos do sistema ou aos zeros de FTs. Por sua vez, os polos e zeros estão associados às ressonâncias paralela e série, respectivamente, de uma FT de interesse do sistema. Estas associações permitem que o desempenho harmônico do sistema possa ser melhorado pelo deslocamento apropriado destes polos e zeros no plano complexo. Este deslocamento pode ser feito de forma eficiente utilizando as sensibilidades deste polo e zeros em relação a componentes do sistema elétrico de potência como, por exemplo, bancos de capacitores (Varricchio and Gomes Jr 2018), (Varricchio et al. 2003).

2.3.2 Matriz Inversa e Matriz Pseudo-Inversa

Uma matriz quadrada $\mathbf{A}$ é dita invertível se existe uma matriz $\mathbf{A^{-1}}$ tal que $\mathbf{AA^{-1}} = \mathbf{A^{-1}A} = \mathbf{I}$, onde $\mathbf{I}$ é a matriz identidade de mesma dimensão de $\mathbf{A}$ (Hoffmann and Kunze 1971). Uma das formas mais comuns e eficientes de cálculo da matriz inversa é a eliminação de Gauss-Jordan, baseada na fatoração **LU** (ver subitem a seguir) (Strang 2013). Deve-se observar que não é eficiente numericamente resolver um sistema de equações lineares $\mathbf{Ax} = \mathbf{b}$ por meio da inversão explicita da matriz $\mathbf{A}$. Ao invés disto, deve-se transformar o sistema original em dois

sistemas triangulares: $\mathbf{Lc} = \mathbf{b}$ e $\mathbf{Ux} = \mathbf{c}$, onde $\mathbf{A} = \mathbf{LU}$. Portanto, a inversão numérica de matrizes não tem muitas aplicações práticas em sistemas de potência, embora possua grande importância teórica e conceitual.

Uma possível generalização da matriz inversa (Penrose 1955) para matrizes retangulares é a matriz $\mathbf{A}^{\dagger}$ chamada de matriz pseudo-inversa (ou inversa de Moore-Penrose). Uma das maneiras de obter a matriz $\mathbf{A}^{\dagger}$, a partir da matriz **A**, é a através de uma decomposição em valores singulares (Trefethen and Bau 1997), (Lord et al. 1999) (Lord et al. 1999) (SVD - *Singular Value Decomposition*), a qual será mencionada mais adiante. De maneira geral, a utilidade desta matriz pseudo-inversa é produzir uma solução de mínimos quadrados (Björck 1996) para um sistema sobredeterminado (mais equações do que incógnitas) de equações lineares. A solução de mínimos quadrados é empregada no método de Ajuste Vetorial (*Vector Fitting* em inglês) (Bjørn Gustavsen and Semlyen 1999). Este método é amplamente utilizado pelos engenheiros de sistemas de potência para a construção de equivalentes de redes elétricas dependentes da frequência (Campello et al. 2020), (Bjorn Gustavsen and De Silva 2013) e modelos de linhas de transmissão (Bjorn Gustavsen 2017), (Bjørn Gustavsen 2006), (Bjørn Gustavsen and Nordstrom 2008) e transformadores (Bjorn Gustavsen 2016), (Bjørn Gustavsen 2003), (Bjørn Gustavsen 2010), entre outras aplicações. Estes equivalentes e modelos são fundamentais para a análise precisa e confiável de transitórios eletromagnéticos.

2.3.3 Fatoração LU

A fatoração ou decomposição LU (*Lower/Upper*) é uma forma de escrever uma matriz **A** não-singular como o produto de duas matrizes triangulares: uma inferior **L** e outra superior **U**. Com estes fatores é possível transformar o sistema original $\mathbf{Ax} = \mathbf{b}$ em dois sistemas triangulares dados por $\mathbf{Lc} = \mathbf{b}$ e $\mathbf{Ux} = \mathbf{c}$. Esta transformação é uma das formas mais eficientes de solução numérica de sistemas de equações lineares (Strang 2013). Esta fatoração possui diversas aplicações em sistemas elétricos de potência, como cálculo de respostas em frequência e análise de contingências (Varricchio, Costa, and Véliz 2015) em estudos de comportamento harmônico, cálculo de respostas no tempo na análise de transitórios eletromecânicos (Daniel et al. 2017) e eletromagnéticos (Daniel et al. 2019), (Daniel 2018).

2.3.4 Decomposições QR, QZ e Valores Singulares

A decomposição ou fatoração QR (Lord et al. 1999) permite escrever uma matriz **A** como o produto de uma matriz ortogonal **Q** por uma matriz triangular superior **R**, usada frequentemente na solução de problemas de mínimos quadrados linear. Também é a base para um método bastante consagrado de cálculo computacional de autovalores chamado de Algoritmo **QR**. Na modelagem de sistemas de potência por equações de estados, este algoritmo permite o cálculo simultâneo de todos os polos do sistema, resíduos associados e zeros de FTs. Estes polos e resíduos são utilizados na análise de estabilidade a pequenas perturbações de sistemas elétricos de potência (Kundur 2017). Na modelagem de sistemas de potência por sistemas descritores (Freitas et al. 2011), (Varricchio and Gomes Jr 2018) a decomposição QZ (Moler and Stewart 1973) é útil no cálculo simultâneo de todos os seus polos, resíduos associados e zeros de FTs. Como já mencionado no subitem 2.3.1, o desempenho harmônico do sistema pode ser melhorado pelo deslocamento apropriado destes polos e zeros no plano complexo (Varricchio and Gomes Jr 2018), (Varricchio et al. 2003).

A SVD (Trefethen and Bau 1997), (Lord et al. 1999), (Horn and Johnson 2012) é útil no cálculo da pseudo-inversa, no ajuste (*fitting*) de funções por mínimos quadrados, na aproximação de matrizes, e na determinação do posto, imagem e núcleo de uma determinada matriz. Esta decomposição é utilizada, dentre outras aplicações, no *Square Root Balanced Truncation Method* para a construção eficiente de modelos reduzidos de sistemas de potência (Varricchio, Freitas, and Martins 2015), (Schilders et al. 2008).

2.3.5 Jacobiana e Hessiana

Seja $F: \mathbb{R}^n \to \mathbb{R}^m$ uma função com domínio n-dimensional e imagem m-dimensional, definida por um vetor de m componentes, onde cada componente é uma outra função $F_i: \mathbb{R}^n \to \mathbb{R}$ cujas derivadas parciais podem ser organizadas em uma matriz $m \times n$ que é chamada de matriz Jacobiana (Sauer and Pai 1990). A matriz Hessiana de uma determinada função de n variáveis é uma matriz quadrada de dimensão $n \times n$ composta pelas derivadas parciais de segunda ordem da função, sendo útil na descrição da curvatura local da função. Em sistemas de potência, estas matrizes são utilizadas na solução do problema de fluxo de potência ótimo (Sasson et al. 1973), (Granville 1994).

3. COMPARAÇÕES MATLAB VERSUS PYTHON

A seguir serão apresentadas as comparações entre Matlab e Python para alguns cálculos e algoritmos úteis na análise de sistemas de potência.

3.1 Autovalores

Para calcular os autovalores de uma matriz com o Python, utilizou-se o pacote "NumPy" (Klein 2014), o qual suporta a utilização de vetores e matrizes multidimensionais, possuindo uma grande quantidade e variedade de funções matemáticas para se trabalhar com tais estruturas. Inicialmente, criou-se no Python uma matriz randômica de dimensão 100×100 utilizando-se a função "random.rand" e, em seguida, calculou-se os autovalores com a função "linalg.eigvals". A mesma matriz foi utilizada no Matlab para calcular os autovalores através da função "eig". Na Figura 1 está apresentado o resultado da comparação no plano complexo, onde verifica-se que houve elevado grau de coincidência entre os resultados. Os erros percentuais obtidos para cada autovalor λ estão mostrados na Figura 2. Este erro é dado por:

$$\epsilon = \left| \frac{\lambda^{Matlab} - \lambda^{Python}}{\lambda^{Matlab}} \right| 100\% \qquad (1)$$

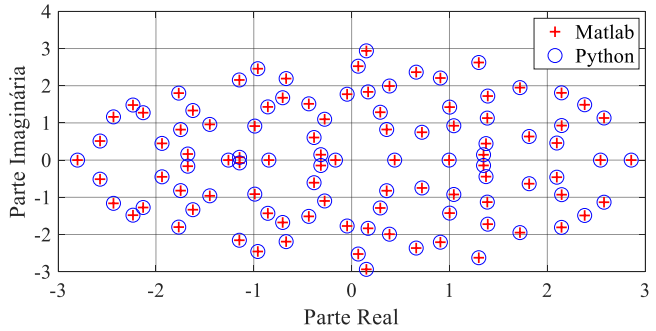

Figura 1 – Comparação de cálculo de autovalores

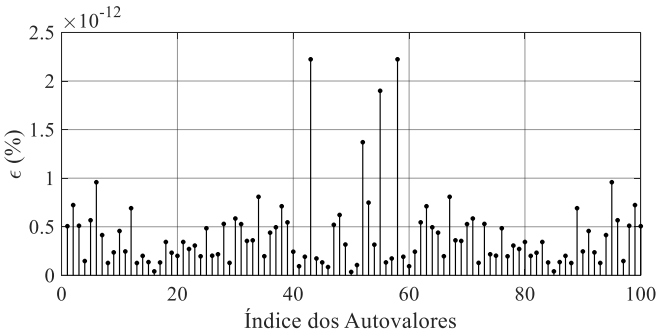

Figura 2 – Erro entre os cálculos de autovalores

Objetivando verificar a propagação do acúmulo dos erros de arredondamento, inerentes aos processos numéricos, foi realizada a mesma comparação anterior, porém para uma matriz randômica de dimensões 1000×1000. Neste caso, o maior erro encontrado foi de 2.24×10^{-11} %.

3.2 Fatoração LU

Para comparar o cálculo da fatoração LU realizado no Python e no Matlab, criou-se um código no Python que gera uma matriz **A** randômica, calcula as matrizes **L** (*Lower*) e **U** (*Upper*), utilizando a função "scipy.linalg.lu", e exporta estas matrizes para um arquivo no formato "MAT-file" (padrão de importação/exportação de variáveis do Matlab). O pacote utilizado "SciPy" possui um grande conjunto de ferramentas, como algoritmos de integração numérica, otimização, interpolação, transformadas, estatística, dentre outras. Em seguida, a mesma matriz **A** foi utilizada no Matlab e calculada a fatoração LU através do comando "lu". Os máximos erros percentual obtidos para cada uma das matrizes triangulares foram calculados conforme (2) e (3).

$$\epsilon_{\mathrm{L}} = \frac{\max\{|\mathbf{L}^{Matlab} - \mathbf{L}^{Python}|\}}{\left|\mathbf{L}_{i,j}^{Matlab}\right|} = 2.95 \times 10^{-12}\ \% \qquad (2)$$

$$\epsilon_{\mathrm{U}} = \frac{\max\{|\mathbf{U}^{Matlab} - \mathbf{U}^{Python}|\}}{\left|\mathbf{U}_{m,n}^{Matlab}\right|} = 6.3 \times 10^{-13}\ \% \qquad (3)$$

Onde $(i,j) = (97{,}96)$ representa a posição (linha, coluna) do elemento $\mathbf{L}_{i,j}$ onde se obteve a maior diferença entre os resultados do Matlab e do Python no cálculo da matriz **L**. Esta posição está mostrada na Figura 3 (ponto de interseção entre as retas horizontal e vertical). Analogamente, $(m,n) = (84{,}93)$ representa a posição (linha, coluna) do elemento $\mathbf{U}_{m,n}$ onde se obteve a maior diferença entre os resultados do Matlab e do Python no cálculo da matriz **U** . Esta posição está mostrada na Figura 4 (ponto de interseção entre as retas horizontal e vertical).

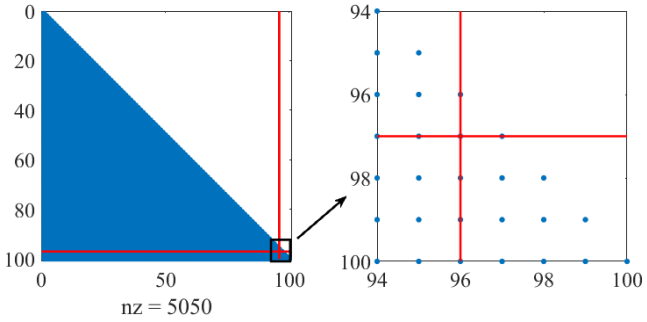

Figura 3 – Posição relativa ao máximo erro na matriz **L**

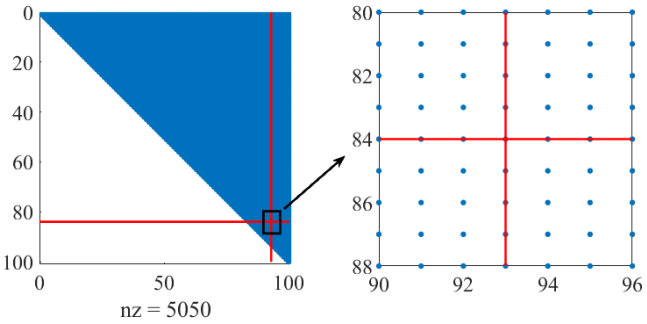

Figura 4 – Posição relativa ao máximo erro na matriz **U**.

3.3 Matemática Simbólica

No Matlab é possível realizar diversas operações com variáveis e funções simbólicas através do objeto "sym". No Python também é possível utilizar matemática simbólica através da biblioteca "SymPy" (Meurer et al. 2017). Através da mesma, é possível fazer desde aritmética básica até cálculos mais avançados como limites, derivadas, integrais, solução de equações polinomiais, algébricas e diferenciais. A seguir, é apresentado um exemplo do cálculo de uma equação diferencial no modo interativo do Python usando símbolos:

```
>>> from sympy import Symbol, Function, Eq, dsolve, sin, diff
>>> x = Symbol("x")
>>> f = Function("f")
>>> eq = Eq(x**2*f(x).diff(x), -3*x*f(x) + sin(x)/x)
>>> eq
 2  d                      sin(x)
x ⋅ ──(f(x)) = -3⋅x⋅f(x) + ──────
    dx                       x
>>>
>>> dsolve(eq, f(x))
        C1 - cos(x)
f(x) = ─────────────
                3
               x
```

3.4 Quadratura Numérica

Para exemplificar o cálculo de integração numérica, seja a seguinte função gaussiana:

$$f(x) = \frac{1}{\sqrt{\pi}} e^{-x^2} \ \mid \ x \in \mathbb{R} \qquad (4)$$

Utilizando-se o método de quadratura Gauss-Kronrod (Laurie 1997) no Matlab, através da função "quadgk", para calcular a área sob a curva de $f(x)$ no intervalo $0 \leq x \leq 1$, obtém-se o valor 0.421350396474857 (com erro absoluto de 2.11636×10^{-17}). Utilizando-se o método de quadratura Gauss-Legendre (Golub and Welsch 1969) no Python, através da função "scipy.integrate.quadrature", para calcular a mesma área anterior, obtém-se 0.421350396518711 (com erro absoluto de 3.45505×10^{-9}).

3.5 Decomposição em Valores Singulares

Para exemplificar a SVD utilizou-se no Matlab a função "svd" e no Python a "numpy.linalg.svd" aplicada a uma matriz randômica de dimensão 100×100. Os resultados de ambos programas foram praticamente os mesmos. Os erros entre os cálculos dos valores singulares estão mostrados na Figura 5. Estes erros foram calculados utilizando equação análoga a (1).

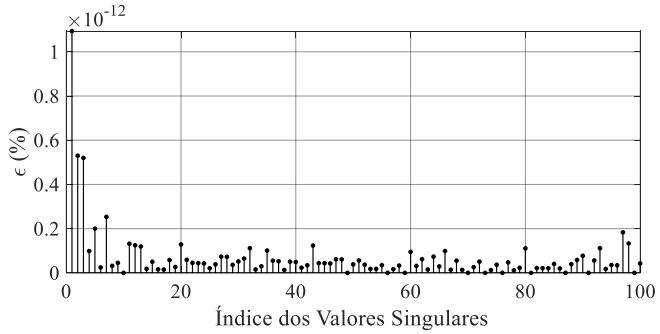

Figura 5 – Erros entre os cálculos dos valores singulares.

3.6 Transformada Rápida de Fourier

Seja a função $v(t)$ do tempo ($t \in \mathbb{R}$) definida em (5), que pode representar, por exemplo, uma tensão com frequência fundamental de 60 Hz ($\omega = 2\pi 60$ rad/s). A função $v_h(t)$, definida em (6), representa as componentes harmônicas de $v(t)$, mostrada na Figura 6.

$$v(t) = \text{sen}(\omega t) + v_h(t) \tag{5}$$

$$\begin{aligned} v_h(t) = {} & 0.05\text{sen}(3\omega t) + 0.045\text{sen}(6\omega t) \\ & + 0.04\text{sen}(7\omega t) + 0.044\text{sen}(11\omega t) \\ & + 0.035\text{sen}(18\omega t) + 0.025\text{sen}(24\omega t) \\ & + 0.01\text{sen}(100\omega t) + 0.007\text{sen}(200\omega t) \\ & + 0.003\text{sen}(300\omega t) \end{aligned} \tag{6}$$

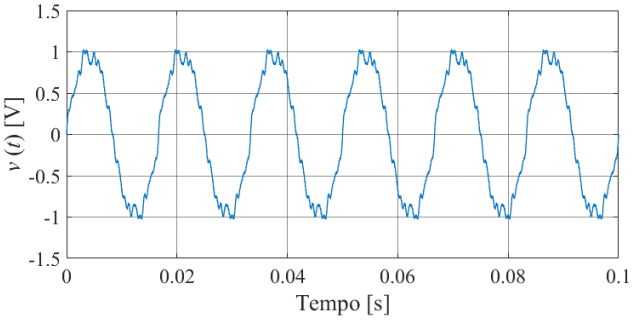

Figura 6 – Forma de onda da tensão $v(t)$.

Na Figura 7 é apresentada a comparação dos resultados obtidos no cálculo da FFT de $v(t)$ (função "fft" no Matlab e função "numpy.fft.fft" no Python). As curvas são visualmente coincidentes e o maior erro relativo encontrado foi de 5.0349×10^{-6} %.

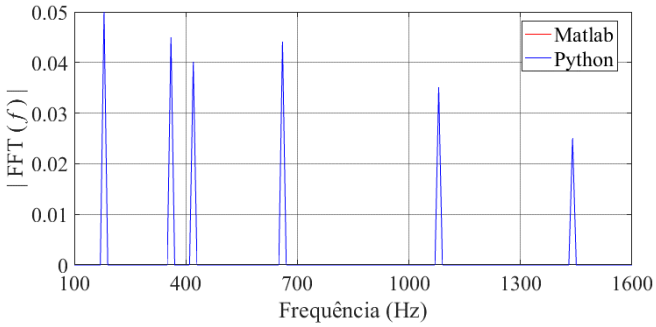

Figura 7 – Conteúdo harmônico predominante de $v(t)$.

3.7 Simulink, PyPSA e Pandapower

O Simulink é uma ferramenta integrada ao Matlab bastante útil para modelagem, análise e simulação de sistemas dinâmicos, incluindo os sistemas de potência (Kulkarni and Chile 2011), (Dessaint et al. 1999). Sua interface é baseada em diagramas de blocos. Este simulador pode ser aplicado tanto para sistemas lineares como não-lineares. As equações algébricas e diferenciais, que descrevem o comportamento dos sistemas a partir dos blocos no Simulink, são solucionadas no domínio do tempo através de alguns métodos de integração disponíveis como o trapezoidal, Euller, Runge-Kutta, dentre outros. O Matlab também possui uma *toolbox* específica para simulação de sistemas de potência, com diversos modelos e recursos disponíveis.

O PyPSA (Brown et al. 2018) é uma *toolbox* gratuita para simulação e otimização de sistemas de potência, que inclui recursos como modelos de unidades geradoras convencionais, de geração eólica e solar, de unidades de armazenamento e acoplamento com planejamento energético e redes mistas de corrente alternada e contínua. O PyPSA foi projetado trabalhar com redes de grande porte e longas séries temporais.

O Pandapower é uma ferramenta baseada em Python para análise e otimização de sistemas de potência balanceados. Ele permite análises de fluxo de potência, fluxo de potência ótimo, estimação de estados e curto-circuito. Ele inclui uma solução de fluxo de potência baseada no método de Newton-Raphson, que utiliza compilação *just-in-time*, para melhorar o desempenho computacional.

4. RECURSOS GRÁFICOS E DEPURAÇÃO DE ERROS

O Matlab tem um depurador de erros integrado em seu editor de códigos. Alternativamente, também é possível utilizar as funções de depuração como "dbstack", "dbstepin", "dbcont", "dbstepou" e "dbquit". O Python também permite utilização de diversos depuradores através das diversas IDEs (*Integrated Development Environment*) disponíveis em código aberto.

Com relação aos recursos gráficos e de plotagem de resultados, o Matlab possui diversas funções integradas de traçado em duas ou três dimensões (pontos, curvas, superfícies, histogramas, etc.). O Python possui uma importante biblioteca de traçado de gráficos, chamada “matplotlib”, que disponibiliza uma grande variedade de gráficos semelhantes àqueles oferecidos pelo Matlab. Além desta, o Python também possui muitas outras bibliotecas alternativas para este fim como, por exemplo, “Pyplot” e “Pylab”.

5. INTERPRETAÇÃO VERSUS COMPILAÇÃO

Tanto o Python como o Matlab são linguagens de programação interpretadas, ou seja, existe um interpretador que não converte todo o código para linguagem de máquina antes de executá-lo. Ao invés disto, ele executa sequencialmente cada instrução. Ambos, Python e Matlab, permitem a utilização da técnica *Just In Time Compiling*, que resulta numa melhora significativa de performance. Também é possível compilar um código Python gerando um único arquivo executável com o "pyinstaller", "py_compiler", "Py2Exe", "cx_Freeze", dentre outros. Com relação ao Matlab, é também é possível compilar o código fonte utilizando o comando "mcc".

6. CONCLUSÕES

Assim como o Matlab, o Python é uma linguagem com considerável maturidade, pois já existe há três décadas. Além disso, uma de suas maiores atrações é o fato de possuir elevada quantidade de pacotes criados por sua grande e dedicada comunidade de milhões de usuários. O Python também tem se tornado mais eficiente computacionalmente devido a melhorias obtidas no processamento e paralelismo.

Tanto o Python quanto o Matlab suportam técnicas de programação orientada a objetos, o que pode facilitar e tornar mais intuitiva a elaboração das estruturas de dados da rede elétrica e oferecer uma melhor gestão de projetos de maior porte. Além disso, ambas linguagens, originalmente interpretadas, permitem facilmente a compilação de códigos fontes para geração de arquivos executáveis (linguagem de máquina). Um grande atrativo do Python é a gratuidade de suas licenças, enquanto as do Matlab são pagas e relativamente caras, fator importante a ser considerado na escolha da ferramenta de prototipação.

Embora diversos fóruns na internet sobre o Python contenham comparações com o Matlab sob vários aspectos, neste artigo deu-se maior atenção aos mais pertinentes à prototipação de ferramentas para análise de sistemas elétricos de potência. Os resultados apresentados indicam o Python como alternativa viável ao Matlab. Ressalta-se, contudo, que estes resultados são preliminares e, portanto, julga-se que sejam necessários mais testes envolvendo problemas reais de sistemas de potência, considerando não apenas a precisão como também o tempo de CPU. Ressalta-se, contudo, que linguagens de alto nível do tipo Python e Matlab, que possuem diversos recursos matemáticos e gráficos, são extremamente importantes para acelerar o desenvolvimento inicial (protótipo) de métodos, modelos, algoritmos e provas de conceito para sistemas elétricos de potência e equipamentos associados.

REFERÊNCIAS

Ames, W. F., & Brezinski, C. (1993). Numerical recipes in Fortran (The art of scientific computing). *Mathematics and Computers in Simulation*. https://doi.org/10.1016/0378-4754(93)90043-t

Björck, Å. (1996). *Numerical Methods for Least Squares Problems*. *Numerical Methods for Least Squares Problems*. https://doi.org/10.1137/1.9781611971484

Brown, T., Hörsch, J., & Schlachtberger, D. (2018). PyPSA: Python for power system analysis. *Journal of Open Research Software*, *6*(1). https://doi.org/10.5334/jors.188

Campello, T. M., Varricchio, S. L., & Taranto, G. N. (2020). Representation of multiport rational models in an electromagnetic transients program: Networks with lumped and distributed parameters. *Electric Power Systems Research*. https://doi.org/10.1016/j.epsr.2019.106029

Cooley, J. W., & Tukey, J. W. (1965). An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*. https://doi.org/10.2307/2003354

Daniel, L. O. (2018, June 29). *Electromagnetic Transient Simulator Using Dynamic Phasors for Non-Linear Analysis of Electrical Networks with FACTS Equipment*. PhD Thesis,COPPE/UFRJ.

Daniel, L. O., & Gomes Jr, S. (2017, October 22). Amortecimento de Modos Eletromecânicos Utilizando Estabilizadores em Elos HVDC Considerando-se Diferentes Estratégias de Controle. *XXIV SNPTEE, Curitiba - PR*.

Daniel, L. O., Gomes Jr, S., Grander, L., & Lirio, F. L. (2013, October 1). Small Signal Analysis of HVDC Systems Using Computational Program PacDyn. *Colloquium CIGRE, Brasilia-DF*.

Daniel, L. O., Gomes Jr, S., & Watanabe, E. H. (2017, May 21). Utilização de Fasores Dinâmicos para Modelagem de Transitórios Eletromecânicos e Eletromagnéticos. *XVII ERIAC, Ciudad del Este, Paraguay*.

Daniel, L. O., Gomes Jr, S., & Watanabe, E. H. (2019, June 21). Novo Simulador de Transitórios Eletromagnéticos Baseado em Fasores Dinâmicos. *XVIII ERIAC, Foz do Iguaçu - PR*.

Dessaint, L. A., Al-Haddad, K., Le-Huy, H., Sybille, G., & Brunelle, P. (1999). A power system simulation tool based on simulink. *IEEE Transactions on Industrial Electronics*, *46*(6), 1252–1254. https://doi.org/10.1109/41.808019

Freitas, F. D., Martins, N., Varricchio, S. L., Rommes, J., & Veliz, F. C. (2011). Reduced-order transfer matrices from RLC network descriptor models of electric power grids. *IEEE Transactions on Power Systems*. https://doi.org/10.1109/TPWRS.2011.2136442

Golub, G. H., & Welsch, J. H. (1969). Calculation of Gauss quadrature rules. *Mathematics of Computation*. https://doi.org/10.1090/s0025-5718-69-99647-1

Gómez, P., & Uribe, F. (2009). The numerical Laplace transform: An accurate technique for analyzing electromagnetic transients on power system devices. *International Journal of Electrical Power & Energy Systems - INT J ELEC POWER ENERG SYST*, *31*, 116–123. https://doi.org/10.1016/j.ijepes.2008.10.006

Granville, S. (1994). Opiimal reactive dispatch through interior point methods. *IEEE Transactions on Power Systems*. https://doi.org/10.1109/59.317548

Gustavsen, Bjorn. (2016). Wideband Transformer Modeling Including Core Nonlinear Effects. *IEEE Transactions on Power Delivery*. https://doi.org/10.1109/TPWRD.2015.2440446

Gustavsen, Bjorn. (2017). Optimal Time Delay Extraction for Transmission Line Modeling. *IEEE Transactions on Power Delivery*. https://doi.org/10.1109/TPWRD.2016.2609039

Gustavsen, Bjørn. (2003). Application of vector fitting to high frequency transformer modeling. *Measurements*, *3*, 4. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.9606&rep=rep1&type=pdf

Gustavsen, Bjørn. (2006). Improving the pole relocating properties of vector fitting. *IEEE Transactions on Power Delivery*. https://doi.org/10.1109/TPWRD.2005.860281

Gustavsen, Bjørn. (2010). A hybrid measurement approach for wideband characterization and modeling of power transformers. *IEEE Transactions on Power Delivery*. https://doi.org/10.1109/TPWRD.2010.2043747

Gustavsen, Bjorn, & De Silva, H. M. J. (2013). Inclusion of rational models in an electromagnetic transients program: Y-Parameters, Z-Parameters, S-parameters, transfer functions. *IEEE Transactions on Power Delivery*. https://doi.org/10.1109/TPWRD.2013.2247067

Gustavsen, Bjørn, & Nordstrom, J. (2008). Pole identification for the universal line model based on trace fitting. *IEEE Transactions on Power Delivery*. https://doi.org/10.1109/TPWRD.2007.911186

Gustavsen, Bjørn, & Semlyen, A. (1999). Rational approximation of frequency domain responses by vector fitting. *IEEE Transactions on Power Delivery*. https://doi.org/10.1109/61.772353

Hoffmann, K., & Kunze, R. (1971). Linear Algebra Second edition. *Mathematics of Computation*, *15*(75), 407. http://www.jstor.org/stable/2002915?origin=crossref

Horn, R. A., & Johnson, C. R. (2012). *Matrix Analysis*. *Matrix Analysis*. https://doi.org/10.1017/cbo9781139020411

Klein, B. (2014). NumPy. In *Einführung in Python 3*. https://doi.org/10.3139/9783446441514.031

Kulkarni, S. B., & Chile, R. H. (2011). MATLAB / SIMULINK Simulation Tool for Power Systems. *International Journal of Power System Operation and Energy Management*, *1*(2), 33–38.

Kundur, P. S. (2017). Power system dynamics and stability. In *Power System Stability and Control, Third Edition*. https://doi.org/10.4324/b12113

Kythe, P. K., & Schäferkotter, M. R. (2004). *Handbook of computational methods for integration*. *Handbook of Computational Methods for Integration*. https://doi.org/10.1201/9780203490303

Laurie, D. P. (1997). Calculation of Gauss-Kronrod quadrature rules. *Mathematics of Computation*. https://doi.org/10.1090/s0025-5718-97-00861-2

Liu, Y., Jiang, B., Wang, C., & Geng, S. (2011). Power system harmonic analysis based on windowed FFT and wavelet transform. In *DRPT 2011 - 2011 4th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies*. https://doi.org/10.1109/DRPT.2011.5993893

Lord, N., Golub, G. H., & Loan, C. F. Van. (1999). Matrix Computations. *The Mathematical Gazette*. https://doi.org/10.2307/3621013

Mayoral, E. H., López, M. A. H., Hernández, E. R., Marrero, H. J. C., Portela, J. R. D., & Oliva, V. I. M. (2017). Fourier Analysis for Harmonic Signals in Electrical Power Systems. *Fourier Transforms - High-tech Application and Current Trends*, (September). https://doi.org/10.5772/66733

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., et al. (2017). SymPy: Symbolic computing in python. *PeerJ Computer Science*. https://doi.org/10.7717/peerj-cs.103

Moler, C. B., & Stewart, G. W. (1973). An Algorithm for Generalized Matrix Eigenvalue Problems. *SIAM Journal on Numerical Analysis*. https://doi.org/10.1137/0710024

Moreno, P., & Ramirez, A. (2008). Implementation of the numerical Laplace transform: A review. *IEEE Transactions on Power Delivery*. https://doi.org/10.1109/TPWRD.2008.923404

Ogata, K. (2010). *Engenharia de Controle Moderno*. *Control Engineering*.

Oppenheim, A. V, & Schafer, R. W. (1998). *Discrete Time Signal Processing 2nd Edition*. *Book*.

Penrose, R. (1955). A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*. https://doi.org/10.1017/S0305004100030401

Poularikas, A. D., & Seely, S. (2010). Laplace transforms. In *Transforms and Applications Handbook, Third Edition*. https://doi.org/10.1201/9781420066531

Rehman, B., Ahmad, M., & Hussain, J. (2015). Analysis of power system harmonics using singular value decomposition, least square estimation and FFT. In *2014 International Conference on Energy Systems and Policies, ICESP 2014*. https://doi.org/10.1109/ICESP.2014.7346973

Sasson, A. M., Viloria, F., & Aboytes, F. (1973). Optimal load flow solution using the Hessian matrix. *IEEE Transactions on Power Apparatus and Systems*. https://doi.org/10.1109/TPAS.1973.293590

Sauer, P. W., & Pai, M. A. (1990). Power system steady-state stability and the load-flow jacobian. *IEEE Transactions on Power Systems*. https://doi.org/10.1109/59.99389

Schilders, W. H. a, Vorst, H. a Van Der, & Rommes, J. (2008). *Model Order Reduction: Theory, Research Aspects and Applications*. *Methods*. https://doi.org/10.1007/978-3-540-78841-6

Segundo-Ramirez, J., Bayo-Salas, A., Esparza, M., Beerten, J., & Gómez, P. (2020). Frequency Domain Methods for Accuracy Assessment of Wideband Models in Electromagnetic Transient Stability Studies. *IEEE Transactions on Power Delivery*, *35*(1), 71–83. https://doi.org/10.1109/TPWRD.2019.2927171

Strang, G. (2013). Linear Algebra and its applications fourth edition. *Pressure Vessel Design Manual*. https://doi.org/10.1016/B978-0-12-387000-1.01001-9

Trefethen, L. N., & Bau, D. (1997). *Numerical Linear Algebra*. *Numerical Linear Algebra*. https://doi.org/10.1137/1.9780898719574

Varricchio, S. L., Costa, C., & Véliz, F. C. (2015, October 18). Método de Alto Desempenho Computacional para Estudos de Impacto Harmônico de Novos Acessantes à Rede Básica. *XXIII SNPTEE, Foz do Iguaçu - PR*.

Varricchio, S. L., Freitas, F. D., & Martins, N. (2015). Hybrid modal-balanced truncation method based on power system transfer function energy concepts. *IET Generation, Transmission and Distribution*. https://doi.org/10.1049/iet-gtd.2014.1116

Varricchio, S. L., Freitas, F. D., Martins, N., & Veliz, F. C. (2015). Computation of dominant poles and residue matrices for multivariable transfer functions of infinite power system models. *IEEE Transactions on Power Systems*. https://doi.org/10.1109/TPWRS.2014.2336243

Varricchio, S. L., & Gomes Jr, S. (2018). Electrical network dynamic models with application to modal analysis of harmonics. *Electric Power Systems Research*. https://doi.org/10.1016/j.epsr.2017.09.016

Varricchio, S. L., Martins, N., & Lima, L. T. G. (2003). A Newton-Raphson method based on eigenvalue sensitivities to improve harmonic voltage performance. *IEEE Transactions on Power Delivery*. https://doi.org/10.1109/TPWRD.2002.806682

Yong, M., & Bolin, W. (2009). The optimization algorithm of reducing dimensional FFT for harmonics analysis of power system. In *1st International Conference on Sustainable Power Generation and Supply, SUPERGEN '09*. https://doi.org/10.1109/SUPERGEN.2009.5348321