Revista Sergipana de Matemática e Educação Matemática

https://doi.org/10.34179/revisem.v9i1.16811

CONSTRUÇÃO DE MODELOS MATEMÁTICOS COM O PYTHON: UMA MOTIVAÇÃO PARA O ESTUDO DE CÁLCULO DIFERENCIAL E LÓGICA MATEMÁTICA

CONSTRUCTION OF MATHEMATICAL MODELS WITH THE PYTHON: A MOTIVATION FOR THE STUDYING DIFFERENTIAL CALCULUS AND MATHEMATICAL LOGIC

Angelo Antunes da Rocha Silva
Universidade Federal Rural de Pernambuco – UFRPE
angelo.antunes.09@gmail.com

Maria Ângela Caldas Didier
Universidade Federal Rural de Pernambuco – UFRPE
maria.didier@ufrpe.br

Silvio Cavalcanti Bonfim
Universidade Federal Rural de Pernambuco – UFRPE
silviocavalcanti2011@gmail.com

Resumo

Este artigo tem como objetivo abordar a construção de modelos matemáticos usando *Python*, fornecendo recursos motivadores para o estudo de cálculo diferencial e lógica matemática. Apresentamos um minicurso chamado "*Python* e Aplicações Matemáticas" em 2021, oferecendo uma perspectiva alternativa para o estudo desses conceitos. Neste trabalho discorremos sobre os tópicos trabalhados no minicurso. Utilizamos as técnicas de ajuste linear de curvas e equações diferenciais ordinárias para ilustrar propriedades de funções reais e suas derivadas, fundamentais no cálculo diferencial. Como exemplo da aplicação de ajuste de curvas, discutimos um modelo que relaciona o comprimento por idade das tilápias do rio Nilo e um modelo de juros de cartão de crédito em um mês. Também abordamos a modelagem por equações diferenciais, incluindo o modelo clássico de crescimento populacional exponencial proposto por Thomas Robert Malthus e o modelo epidemiológico SIR (Suscetível-Infectado-Recuperado), desenvolvido por Kermack e McKendrick em 1927. Apresentamos o passo a passo da construção desses modelos e exibimos seus gráficos usando *Python* no ambiente *Google Colaboratory*, permitindo visualizar as estruturas de lógica matemática da linguagem. Além do exposto, disponibilizamos sugestões de atividades para o ensino de Matemática e fornecemos um *link* de acesso às resoluções e códigos dos autores.
Palavras-chave: Modelos Matemáticos, Ajuste Linear de Curvas, Equações Diferenciais, *Python*, Cálculo Diferencial, Lógica Matemática.

Abstract

This article has as objective approach the construction of mathematical models using Python, providing motivational resources for the study of differential calculus and mathematical logic. We produced a short course called "Python e Aplicações Matemáticas" in 2021, offering an alternative perspective for study of these concepts. In this work, we discuss the topics covered in the short course. Using the techniques of curve fitting and ordinary differential equations for we illustrate properties of real functions and their derivatives, which are fundamental to differential calculus. As an example of curve fitting application, we discussed a model that relates the size of tilapias from the Nile River to their age, and another model that relates credit card fees. We also discuss models with differential equations, including the classical model of exponential population growth proposed by Thomas Robert Malthus and the epidemiological SIR model (Susceptible-Infected-Recovered), developed by Kermack and McKendrick in 1927. We present a step-by-step construction of these models and display their graphics using Python in the Google Colaboratory environment, which allows us to visualize the mathematical logic of the language. In addition to what was mentioned, we offer exercise suggestions for teaching mathematics and provide a link to access the solutions and codes created by the authors.
Keywords: Mathematical Models, Curve Fitting, Differential Equations, Python, Differential Calculus, Mathematical Logic.

INTRODUÇÃO

No início do curso de Licenciatura em Matemática da UFRPE, os alunos têm acesso a disciplinas que exploram os conceitos e as propriedades das funções reais e suas derivadas, sendo fundamentais para o estudo do cálculo diferencial. Além disso, eles também aprendem sobre lógica matemática, o que se mostra crucial para a utilização de *softwares* nas pesquisas de Matemática Aplicada. No entanto, é importante ressaltar que os estudantes encontram muita dificuldade em compreender esses assuntos, o que resulta em reprovação nas disciplinas relacionadas e, em alguns casos, até mesmo na desistência do curso.

Diante dessa situação, com o intuito de propor o ensino-aprendizagem dessas disciplinas sob a perspectiva da matemática aplicada, no ano de 2021, decidimos criar uma atividade que envolvesse a Modelagem Matemática e a linguagem de programação *Python*.

De acordo com Bassanezi, a Modelagem Matemática:

> É uma forma de abstração e generalização com a finalidade de previsão de tendências. A modelagem consiste, essencialmente, na arte de transformar situações da realidade em problemas matemáticos cujas soluções devem ser interpretadas na linguagem usual. (BASSANEZI, 2002, p. 24)

Segundo Stewart (2017), quando descrevemos matematicamente um fenômeno do mundo real, podemos utilizar uma função ou uma equação, que é conhecida como modelo

matemático. O propósito desse modelo é explicar o fenômeno em questão e, potencialmente, fazer previsões sobre seu comportamento futuro.

Em várias áreas científicas, são utilizados modelos matemáticos para investigar questões como o crescimento populacional, as dinâmicas entre presas e predadores, a poluição de rios, a absorção de substâncias químicas no organismo e a dinâmica de transmissão de doenças epidêmicas. Durante a pandemia da COVID-19, esses modelos foram amplamente divulgados pela mídia como uma ferramenta para acompanhar a evolução da doença. Um exemplo de um desses modelos pode ser encontrado no estudo conduzido por Tang (2021).

A construção de modelos matemáticos envolve a utilização de técnicas clássicas, como a Regressão ou Ajuste de Curvas com o Método dos Mínimos Quadrados (RUGGIERO & LOPES, 2000), e as Equações Diferenciais Ordinárias (BOYCE & DIPRIMA, 2010). Cada uma dessas técnicas tem suas próprias características e a escolha da abordagem adequada depende do pesquisador. Em alguns casos, especialmente quando lidamos com sistemas de equações diferenciais ou grandes volumes de dados, é necessário recorrer ao uso de métodos numéricos implementados em computadores para obter soluções, pois nem sempre há soluções analíticas diretas disponíveis.

No que diz respeito à Modelagem Matemática, a revista apresentou um artigo que aborda um modelo matemático para calcular a média mensal da evapotranspiração de referência para a cidade de Januária (MG) (MACÊDO et al., 2021), publicado no mesmo ano, e outro artigo que expõe um modelo para previsão de jogos de futebol (RAMOS et al., 2021), publicado em 2021, disponível no volume 6, número 1 da revista.

De acordo com Norvig (1994), cientista da computação e diretor de pesquisa em inteligência artificial na *Google*, é importante ter conhecimentos em lógica matemática para aprimorar as habilidades de programação e resolução de problemas. Em seu livro "Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp", ele explora a utilização de princípios lógicos na programação e enfatiza como o domínio da lógica matemática pode elevar a qualidade do código escrito.

Em 2020, a Revista Sergipana de Matemática e Educação Matemática (ReviSeM) publicou seu primeiro artigo sobre o uso de programação no ensino da Matemática com a linguagem Scratch (ROCHA et al., 2021), bem como um segundo trabalho sobre o

pensamento computacional e o uso de atividades de programação no ensino da Matemática (PRADO et al., 2021). Ambos os artigos podem ser encontrados no volume 5, número 2 da revista.

Python é uma linguagem de programação desenvolvida por Guido van Rossum, um matemático e programador holandês nascido em Haarlem, em 31 de janeiro de 1956 e graduado pela Universidade de Amsterdã (GUIDO, 2021). Com o *Python*, é possível utilizar bibliotecas que oferecem funcionalidades para o desenvolvimento de projetos e a implementação de aplicações complexas. É importante destacar que, assim como outras linguagens de programação, o *Python* é baseado em lógica matemática e segue uma estrutura lógica.

Com base no conhecimento sobre Modelagem Matemática e considerando as oportunidades oferecidas pela linguagem *Python* na construção de modelos matemáticos alinhados ao interesse dos estudantes em tecnologia e suas aplicações, desenvolvemos um minicurso intitulado "*Python* e Aplicações Matemáticas". Esse curso está vinculado ao projeto de ensino denominado "Ciclo de Atividades Complementares do Curso de Licenciatura em Matemática da UFRPE".

Neste trabalho, são exploradas as técnicas de ajuste linear de curvas e equações diferenciais ordinárias, com exemplos práticos para ilustrar o processo de ajuste. Discutimos um modelo que relaciona o comprimento por idade das tilápias do rio Nilo, apresentado por Bassanezi (2002), bem como um segundo modelo que descreve os juros de um cartão de crédito em um período de um mês, conforme encontrado em Silva (2021). Além disso, abordamos a modelagem por meio de equações diferenciais, incluindo o modelo clássico de crescimento populacional exponencial proposto por Thomas Robert Malthus e o modelo epidemiológico compartimental SIR, desenvolvido em 1927 por Kermack e McKendrick, para descrever a dinâmica de transmissão de doenças infecciosas em uma população. Esses modelos são referenciados em Bassanezi (2002). Em seguida, detalhamos os códigos produzidos em *Python* no ambiente do *Google Colaboratory*, fornecendo um passo a passo para gerar os gráficos correspondentes a esses modelos. Adicionalmente, oferecemos uma seção com sugestões de atividades para o ensino de Matemática, além de um *link* para acessar as resoluções e os códigos desenvolvidos pelos autores deste artigo utilizando *Python*.

AJUSTE DE CURVAS E MODELOS

De acordo com Ruggiero e Lopes (2000), o *método de ajuste de curvas ou método de Regressão* é utilizado para encontrar uma curva que se ajuste a um conjunto de pontos e que possivelmente cumpre uma série de hipóteses adicionais. Este método propõe estabelecer e/ou verificar uma correlação entre uma variável dependente e outra independente após realização de uma coleta de dados. A partir do ajuste é possível encontrar tendências e padrões sobre o fenômeno estudado, fazer conjecturas e até validar o modelo desenvolvido. Existem dois métodos de ajuste, o *linear* e o *não linear*. Neste trabalho, discutiremos o ajuste linear.

Uma forma de encontrar a curva que melhor ajusta os dados, isto é, a que minimize a soma dos quadrados das diferenças entre os valores reais e os valores previstos pela curva é através do método que descrevemos abaixo.

Método dos Mínimos Quadrados

O *método dos mínimos quadrados* (MMQ) consiste em encontrar os parâmetros de uma função modelo para que ela ajuste melhor um conjunto de dados representados por pares ordenados. Mas precisamente, considerando *n* pontos (x_i, y_i), $i = 1,2,\ldots,n$, onde x_i é uma variável independente e y_i é uma variável dependente cujo valor é encontrado por observação, a função modelo é pensada como uma fórmula $\varphi(x,\beta)$, onde β é um vetor que mantém os parâmetros ajustáveis. O objetivo é encontrar os valores dos parâmetros que resultam na aproximação mais adequada aos dados disponíveis. O ajuste do modelo é feito por seu resíduo ou desvio, que é definido como a diferença entre o valor real da variável dependente e o valor previsto pelo modelo. Esse resíduo é denotado por r_i, isto é,

$$r_i = y_i - \varphi(x_i,\beta).$$

O MMQ encontra os valores dos parâmetros minimizando a soma dos quadrados residuais expressa da seguinte forma:

$$S = \sum_{i=1}^{n} r_i^2.$$

Destacamos que os dados observados podem ser apresentados de forma *discreta* ou *contínua*. O caso é denominado contínuo quando os dados se apoiam em uma função

contínua definida em Stewart (2017). Nos dois contextos, a função S será minimizada com relação aos parâmetros da função modelo $\varphi(x,\beta)$.

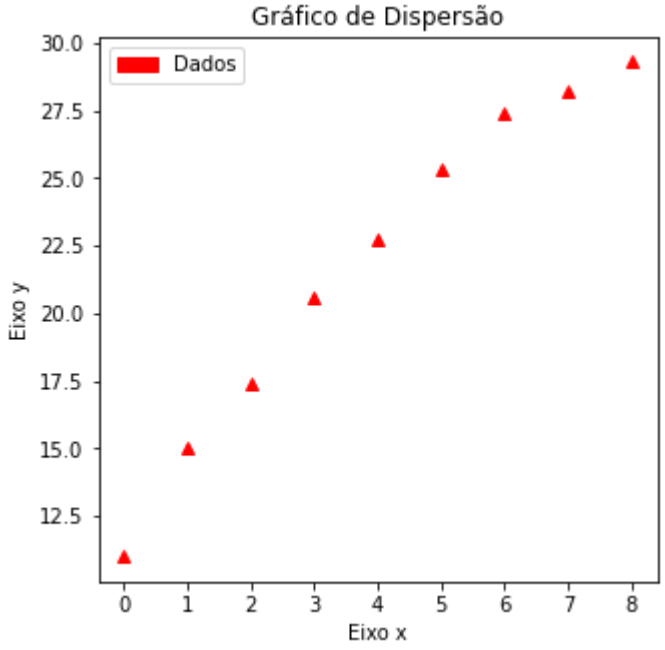
Ajuste Linear de Curvas

Em Ruggiero e Lopes (2000), o *método de ajuste linear de curvas ou regressão linear* é baseado em dados observados da forma $(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))$ com $x_1, x_2, \ldots, x_n \in [a,b]$ e escolhidas m funções $g_1(x), g_2(x), \ldots, g_m(x)$ contínuas em $[a,b]$, fornece m constantes $\beta_1, \beta_2, \ldots, \beta_m$, de modo que:

$$\varphi(x,\beta) = \beta_1 g_1(x) + \beta_2 g_2(x) + \ldots + \beta_m g_m(x)$$

se aproxime ao máximo de $f(x_i)$ para cada valor de x_i, com $i = 1, \ldots, n$. O primeiro passo para se fazer um ajuste de curvas é localizar os dados observados no sistema de coordenadas cartesianas. Esta representação é conhecida como *gráfico de dispersão,* conforme exemplificado na figura a seguir.

Figura 1: Exemplo de um gráfico de dispersão.



Fonte: Autores.

Analisando gráfico de dispersão, escolhemos as funções $g_i(x), i = 1, \ldots, m$. O objetivo é minimizar a soma dos quadrados residuais e, como comentamos anteriormente, realizamos este processo através do MMQ.

Os dados podem ser apresentados de forma discreta ou contínua. No caso contínuo, a função $\varphi(x,\beta)$ de ajuste será a qual proporcionar a menor área entre ela e a função $f(x)$ no intervalo desejado.

De acordo com Gonçalves e Flemming (2007) demonstra-se que se as funções $g_i\,(x), i = 1,\ldots,m$ forem escolhidas de modo que os vetores associados sejam ortogonais, então o sistema admitirá solução única $(\beta_1,\beta_2,\ldots,\beta_m)$ e esta solução será um ponto de mínimo de *S.*

Os modelos a seguir tratam da aplicação do método de regressão linear nos casos discreto e contínuo, respectivamente.

Modelo 1: Comprimento por idade das tilápias do rio Nilo.

A tabela 1 fornece dados sobre as tilápias do rio Nilo, peixe de origem africana e bem-adaptado em águas brasileiras. A seguir apresentaremos o ajuste linear para estes dados.

Tabela 1: Dados sobre idade e comprimento de Tilápias do rio Nilo.

x: Idade	0	1	2	3	4	5	6	7	8
f(x): comprimento	11	15	17.4	20.6	22.7	25.3	27.4	28.2	29.3

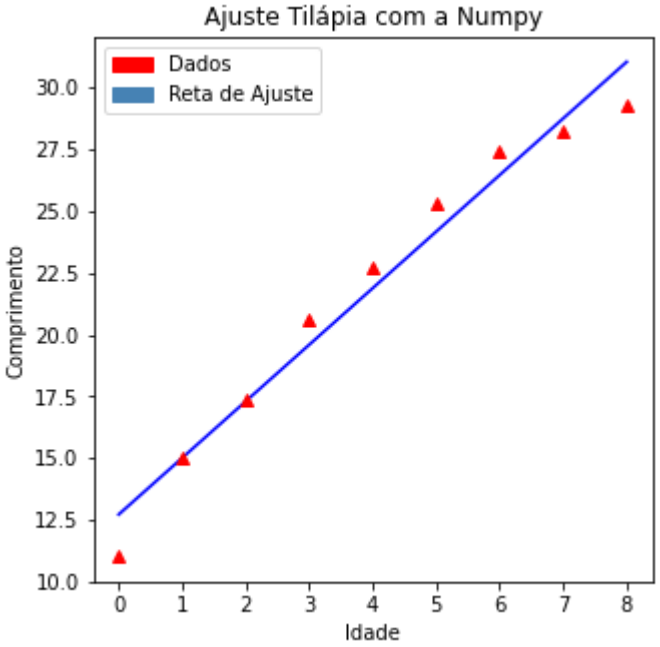
Fonte: Bassanezi (2002).

Resolvendo este problema pelo MMQ para dados discretos, desenvolvido em Silva (2021), encontramos $\beta_1 = 12.71$ e $\beta_2 = 2.291$. Portanto, a equação de reta que ajuste os dados de comprimento e idade das tilápias é dada por:

$$\varphi(x,y) = 12.71 + 2.291x.$$

Segue abaixo o gráfico do ajuste dos dados do problema proposto.

Figura 2: Ajuste de dados do modelo discreto.



Fonte: Autores.

Modelo 2: Juros de um cartão de crédito em um intervalo de um mês.

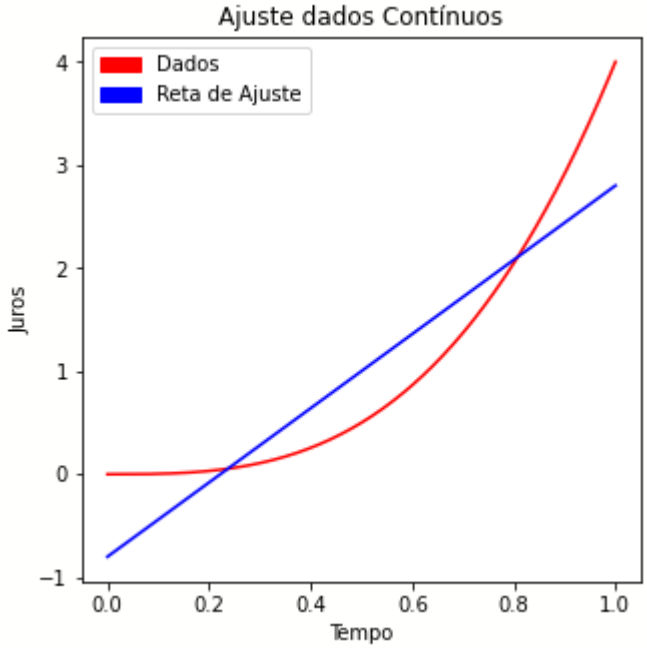
As altas taxas de juros impostas pelos bancos e cartões de crédito estão entre as principais causadoras de endividamento no nosso país. Suponha que os juros de um cartão de crédito pago em reais no primeiro mês é expresso por uma função $f(x) = 4x^3$, em que x refere-se ao tempo. Vamos determinar a função do primeiro grau $\varphi(x) = \beta_2 + \beta_1 x$ que melhor possa descrever estes juros no primeiro mês.

Neste caso, conforme os cálculos apresentados em Silva (2021) usando o MMQ para dados contínuos, obtemos $\beta_1 = 4/5$ e $\beta_2 = 18/5$. Portanto, a reta que representa os juros no primeiro mês é dada por:

$$\varphi(x, \beta) = \frac{18}{5} - \frac{4}{5}x.$$

Abaixo apresentamos o gráfico do ajuste deste exemplo.

Figura 3: Ajuste do modelo contínuo.



Fonte: Autores.

EQUAÇÕES DIFERENCIAIS E MODELOS

Em Boyce e Diprima (2010), uma *equação diferencial* (ED) é uma equação que possui um ou mais termos envolvendo derivadas de uma determinada função. Quando essa equação apresenta derivadas de uma função em uma única variável recebe o nome de *equação diferencial ordinária* (EDO). A partir desta técnica é possível modelar inúmeros problemas de diversas áreas do conhecimento.

De outro modo, uma EDO é uma igualdade envolvendo uma função y, de uma variável independente x, e suas derivadas em relação a essa variável independente. Ou seja, é uma equação da seguinte forma:

$$F[x, y, y', \ldots, y^{(n)}] = 0.$$

A *ordem* de uma EDO é definida pela maior ordem entre as derivadas da função, envolvida na EDO. Outra importante classificação relacionada às equações diferenciais ordinárias diz respeito à linearidade. A EDO é chamada *linear* se F é uma função linear nas variáveis $y, y', y'', \ldots, y^{(n)}$. Portanto, uma EDO linear de ordem n é dada por:

$$a_0(x)y^{(n)} + a_1(x)y^{(n-1)} + \ldots + a_n(x)y = g(x).$$

Caso a EDO não satisfaça a equação anterior, ela é dita *não linear*. Neste trabalho vamos nos restringir às EDOs lineares da forma:

$$y^{(n)}(x) = f[x, y(x), y'(x), \ldots, y^{(n-1)}(x)],$$

sendo f uma função linear das variáveis $y, y', y'', \ldots, y^{(n-1)}$.

Por fim, em particular, uma *solução* de uma equação diferencial ordinária linear em um intervalo (α, β), com α e β números reais, será uma função $\varphi(x)$ tal que $\varphi'(x), \varphi''(x), \ldots, \varphi^{(n)}(x)$ existem e satisfazem

$$\varphi^{(n)}(x) = f[x, \varphi(x), \varphi'(x), \ldots, \varphi^{(n-1)}(x)],$$

para todo $x \in (\alpha; \beta)$. De maneira menos formal, quando resolvemos uma equação diferencial ordinária linear deste tipo, encontramos uma família de funções tal que a derivada destas em um determinado ponto x é imagem de f aplicada em x. A família de todas as curvas que satisfazem essa condição é chamada de *solução geral* dessa EDO linear de ordem n.

Posteriormente, apresentaremos a formulação de um problema que envolve mais de uma equação diferencial ordinária. Nesse caso, é necessário obter uma solução para cada uma dessas equações. Esses problemas são denominados como *sistemas de equações diferenciais ordinárias* (SEDO).

Em particular, quando conhecemos o comportamento da solução $y = \varphi(x)$ de uma EDO de ordem 1 em algum ponto (x_0, y_0), temos um *problema de valor inicial* (PVI). Portanto, um PVI é representado por:

$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}x} = f(x, y), \\ y(0) = y_0. \end{cases}$$

Quando f e $\frac{\partial f}{\partial y}$ (derivada parcial de f em relação a y) são contínuas em um retângulo R: $|x| \leq a, |y| \leq b$ com a e b números reais, temos uma única solução garantida pelo teorema de Picard, contido em Boyce e Diprima (2010).

A seguir abordaremos dois exemplos de modelos matemáticos clássicos. O Modelo de Crescimento Populacional Exponencial de Malthus definido por uma equação diferencial ordinária linear de primeira ordem com condição inicial e o Modelo Epidemiológico do tipo SIR, de Kermack-Mckendrick, definido por um SEDO não linear de primeira ordem também sujeito a condições iniciais. Para a resolução dos mesmos,

assumimos que o leitor conhece o método dos fatores integrantes e o método de equações separáveis, também tratados em Boyce e Diprima (2010).

Modelo 3: Modelo de Crescimento Populacional Exponencial.

O *modelo de crescimento populacional exponencial* (MCPE) observado em Bassanezi (2002) foi desenvolvido pelo britânico Thomas Robert Malthus. Sua teoria ficou conhecida como malthusianismo e tinha como hipótese o argumento de que o crescimento de uma população era proporcional à população em cada instante, isto é, quanto maior a população, maior também será seu crescimento. Dessa forma, a população iria crescer de forma acelerada, sem nenhuma inibição, e assim, em pouco tempo teríamos um ambiente inabitável. Considerando essa teoria, a formulação do modelo é representada pelo sistema abaixo:

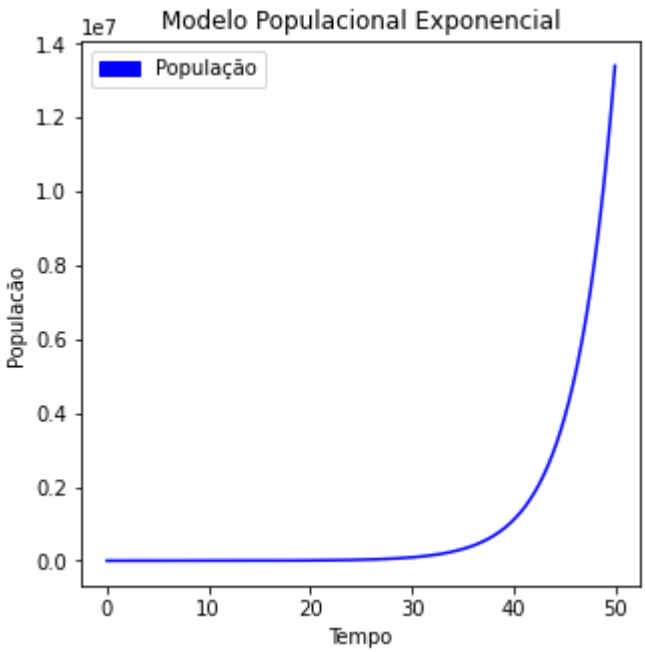
$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}x} = ky, \\ y(0) = y_0 \end{cases}$$

em que k é o coeficiente de crescimento populacional, y a população em cada instante e $y(0) = y_0$ é a população inicial. Note que o crescimento da população y independe do tempo x. A solução analítica do MCPE, encontrada em Silva (2021), é dada por:

$$y = y_0 e^{kx}.$$

Segue abaixo o gráfico da simulação desse modelo considerando 50 unidades de tempo com o coeficiente de crescimento $k = 0.25$ e a população inicial $y(0) = 50$.

Figura 4: Simulação do MCPE.



Fonte: Autores.

Nesse exemplo, podemos observar o impacto no crescimento da população quando ele é modelado por uma função exponencial. Notamos que a medida que o tempo cresce a variação do crescimento a população aumenta mais e mais.

O sociólogo belga Pierre François Verhulst levou em consideração a ideia de que toda população é exposta a sofrer inibições em seu crescimento e assim a quantidade dessa população com o passar do tempo chegaria a um limite máximo. Tal limite é conhecido como *capacidade de suporte.* Esse modelo é denominado *modelo de crescimento populacional logístico* e você encontrará a sua formulação seguida de exemplo em Silva (2021).

Modelo 4: Modelo Epidemiológico Compartimental SIR

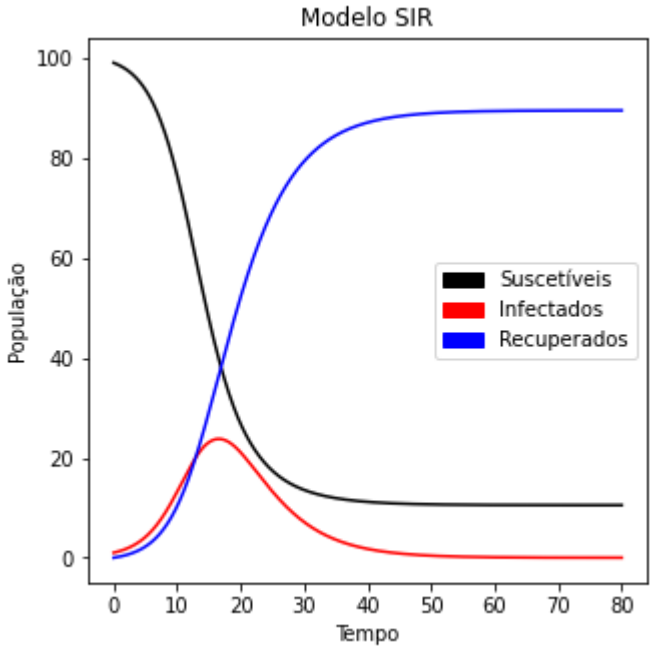
O *modelo epidemiológico compartimental* SIR visto em Bassanezi (2002) foi desenvolvido em 1927 por Kermack e McKendrick, sendo pioneiro nos estudos de doenças transmissíveis epidêmicas. Neste modelo, a população é considerada constante e é dividida em três compartimentos (grupos): *Suscetíveis* (S), composto por indivíduos expostos à contaminação; *Infectados* (I), constituído por indivíduos que foram contaminados e estão transmitindo a doença; e *Recuperados* (R), formado por indivíduos que se recuperaram da doença e adquiriram imunidade. Um indivíduo passa do grupo de suscetível para o dos

infectados a uma taxa α de contágio após entrar em contato com algum infectado, esta dinâmica é descrita em uma primeira equação do sistema pelo termo $-\alpha SI$. Por outro lado, um indivíduo que saiu do grupo de suscetíveis para o dos infectados, é representado pelo termo αSI em uma segunda equação. De maneira semelhante, um indivíduo que deixa o grupo de infectados a uma taxa β de recuperação, tem a dinâmica representada pelo termo $-\beta I$, também na segunda equação. Por fim, um indivíduo que sai do grupo de infectados e entra para o dos recuperados tem esta dinâmica representada pelo termo βI em uma terceira equação. A formulação do modelo é dada por um sistema de equações diferenciais ordinárias de primeira ordem não lineares conhecido como *sistema compartimental não linear*. Eles são formulados apoiados no *princípio da ação das massas* presente em Bassanezi (2002). Este princípio é baseado no encontro das variáveis e a interação entre elas é representada pelo produto entre estas variáveis. A formulação desse modelo compartimental não linear clássico é dada por:

$$\begin{cases} \dfrac{\mathrm{d}S}{\mathrm{d}t} = -\alpha SI, \\ \dfrac{\mathrm{d}I}{\mathrm{d}t} = \alpha SI - \beta I, \\ \dfrac{\mathrm{d}R}{\mathrm{d}t} = \beta I. \end{cases}$$

Observe que estas equações refletem, claramente, que o tamanho da população $N = S(t) + I(t) + R(t)$ *é* constante no tempo. A seguir apresentamos o gráfico que representa os resultados da simulação deste modelo onde foi considerada uma população com um total de 100 indivíduos, sendo inicialmente *99* suscetíveis, 1 infectado e nenhum recuperado. O coeficiente de contágio e a taxa de recuperação foram de, respectivamente, $\alpha = 0.005$ e $\beta = 0.2$ em 80 unidades de tempo.

Figura 5: Simulação do modelo SIR.

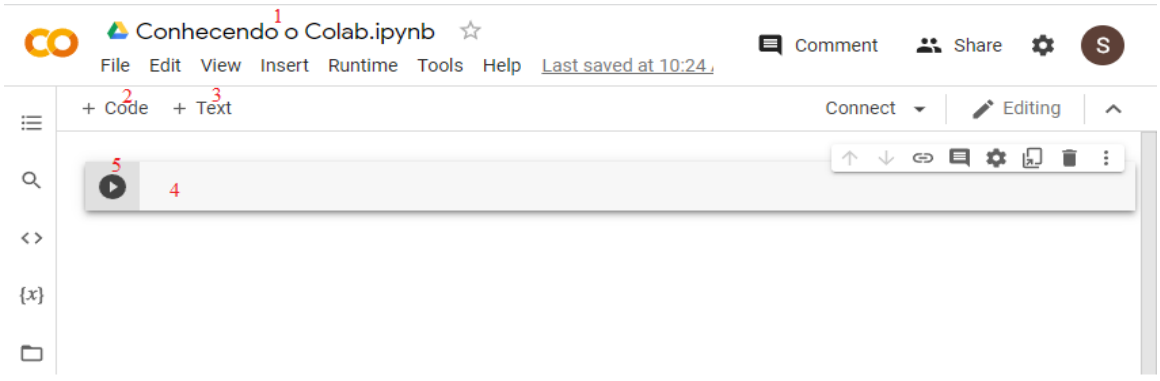


Fonte: Autores.

CÓDIGO DOS MODELOS MATEMÁTICOS EM LINGUAGEM *PYTHON*

A seguir, apresentamos o passo a passo da construção dos modelos matemáticos discutidos nas seções anteriores utilizando *Python*. Esses modelos foram aplicados no minicurso "*Python* e Aplicações Matemáticas", vinculado ao projeto de ensino denominado "Ciclo de Atividades Complementares do Curso de Licenciatura em Matemática da UFRPE". Assumimos que o leitor tem conhecimento prévio desta linguagem e sugerimos que seja utilizada a plataforma *online Google Colaboratory* usando uma conta do *gmail*. A figura abaixo mostra algumas ferramentas e parte desse ambiente.

Figura 6: Plataforma *online Google Colaboratory*.



Fonte: Autores.

1. Nome do arquivo. Importante ressaltar que o termo .ipynb não pode ser apagado;
2. Cria um campo de código;
3. Cria um campo de texto;
4. Campo de código;
5. Executa o código.

Para gerar os códigos descritos abaixo foram utilizadas as bibliotecas: *Numpy* especificidades através das referências (THE NUMPY COMMUNITY, 2021), (SYMPY DEVELOPMENT TEAM, 2021), (SCIPY DEVELOPERS, 2021) e (THE MATPLOTLIB DEVELOPMENT TEAM, 2021). Ressaltamos que no código do modelo 1 foi utilizada a função *polyfit*, que ajusta dados discretos (x,y) através do MMQ. Já no modelo 2 o código foi gerado seguindo o passo a passo do MMQ para o caso contínuo.

Modelo 1: Tilápia idade por comprimento (Linear discreto).

```
import numpy as np #Importa a biblioteca numpy.
import matplotlib.pyplot as plt #Importa o módulo pyplot da biblioteca matplotlib.
import matplotlib.patches as mp #Importa o módulo patches da biblioteca matplotlib.
meidan_age = np.array([0, 1,2, 3, 4, 5, 6, 7, 8])  #Define um vetor cujas entradas são os dados das idades das tilápias.
meidan_size = np.array([11.0, 15, 17.4, 20.6, 22.7,25.3, 27.4, 28.2, 29.3]) #Define um vetor cujas entradas são os dados dos tamanhos das tilápias.
fit = np.polyfit(meidan_age,meidan_size,1) #Ajusta um polinômio de grau 1 para pontos (x, y) e retorna um vetor de coeficientes.
poly = np.poly1d(fit) #Constrói um polinômio a partir de um vetor de coeficientes.
x_poly = np.linspace(meidan_age[0],meidan_age[-1],100)  #Cria um vetor com uma partição do intervalo que vai da primeira à  última entrada do vetor "meidan_age" com 100 pedaços de tamanhos iguais.
y_poly = poly(x_poly) #Aplica o polinômio no vetor com a partição do eixo x.
fig = plt.figure(figsize=(5, 5)) #Define as dimensões  do gráfico em 5 polegadas cada.
plt.plot(x_poly, y_poly, color="blue") #Imprime em cor azul o polinômio resultante do ajuste.
plt.plot(meidan_age,meidan_size,color="red", marker="^", linewidth=0.0) #Imprime no formato de triângulos em cor vermelha os pontos que queremos ajustar.
red_patch = mp.Patch(color= "red", label="Dados") #Constrói legenda para os dados.
steel_blue_patch = mp.Patch(color= "steelblue",label= "Reta de Ajuste") #Constrói
```

```
legenda para o polinômio de ajuste.
plt.legend(handles=[red_patch, steel_blue_patch]) #Adiciona o quadro de legendas no gráfico.
plt.title("Ajuste Tilápia com a Numpy") #Adiciona o título no gráfico.
plt.xlabel("Idade") #Nomeia o eixo x.
plt.ylabel("Comprimento") #Nomeia o eixo y.
plt.show() #Imprime o gráfico.
```

Modelo 2: Juros de um cartão de crédito em um intervalo de um mês (Linear contínuo).

```
import numpy as np #Importa a biblioteca numpy.
import matplotlib.pyplot as plt #Importa o módulo pyplot da biblioteca matplotlib.
import matplotlib.patches as mp #Importa o módulo patches da biblioteca matplotlib.
import sympy as sp #Importa a biblioteca Sympy.
x = sp.Symbol("x") #Define o símbolo que representa a variável para as funções que serão usadas ao recorrer a biblioteca Sympy.
a11 = float(sp.integrate((1), [x,0,1])) #Calcula a integral de f(x)=1 no intervalo [0,1] na variável x e define o resultado como a entrada a11 da matriz A. O comando float está sendo utilizado para converter o resultado simbólico em um número decimal.
a12 = float(sp.integrate(x, [x,0,1])) #Calcula a integral de f(x)=x no intervalo [0,1] na variável x e define o resultado como a entrada a12 da matriz A.
a21 = a12 #Define a entrada a21 igual a entrada a12 da matriz A.
a22 = float(sp.integrate(x**2, [x,0,1])) #Calcula a integral de f(x)=x² no intervalo [0,1] na variável x e define o resultado como a entrada a22 da matriz A.
A = np.array([[a11,a12],[a21,a22]]) #Define a matriz A.
b11 = float(4*sp.integrate(x**3, [x,0,1])) #Calcula a integral de f(x)=4x³ no intervalo [0,1] na variável x e define o resultado como a entrada b11 da matriz B.
b21 = float(4*sp.integrate(x**4, [x,0,1])) #Calcula a integral de f(x)=4x⁴ no intervalo [0,1] na variável x e define o resultado como a entrada b21 da matriz B.
B = np.array([b11,b21]) #Define a matriz B.
S = np.linalg.solve(A, B) #Calcula a solução do sistema na forma matricial A.X=B com primeira entrada correspondendo ao coeficiente angular e a segunda ao termo independente da função do primeiro grau.
poly = np.poly1d(np.flip(S)) #Constrói um polinômio do primeiro grau a partir de flip(S) que é o vetor S reordenado.
```

```
fig = plt.figure(figsize=( 5, 5)) #Define as dimensões do gráfico em 5 polegadas cada.
t = np.linspace(0,1,5000) #Cria um vetor com uma partição do intervalo [0,1] com 5000 pedaços de tamanho igual.
plt.plot(t,4*t**3,color= "red") #Imprime em cor vermelha a curva que descreve os dados.
plt.plot(t,poly(t),color= "blue") #Imprime em cor azul o polinômio resultante do ajuste.
greenpatch = mp.Patch(color= "red", label= "Dados") #Constrói legenda para os dados.
orangepatch = mp.Patch(color= "blue", label= "Reta de Ajuste") #Constrói legenda para a reta de ajuste.
plt.legend(handles=[greenpatch, orangepatch]) #Adiciona o quadro de legendas no gráfico.
plt.title("Ajuste dados Contínuos") #Adiciona o título no gráfico.
plt.xlabel("Tempo") #Nomeia o eixo x.
plt.ylabel("Juros") #Nomeia o eixo y.
plt.show() #Imprime o gráfico.
```

Modelo 3: Modelo de crescimento populacional exponencial (EDO).

```
import numpy as np #Importa a biblioteca numpy.
import matplotlib.pyplot as plt #Importa o módulo pyplot da biblioteca matplotlib.
import matplotlib.patches as mp #Importa o módulo patches da biblioteca matplotlib.
import scipy.integrate as scp #Importa o módulo integrate da biblioteca scipy.
k1 = 0.25 #Taxa de crescimento do modelo MCPE.
time = np.linspace(0, 50, 5000) #Cria um vetor com uma partição do intervalo [0,50] com 5000 pedaços de tamanhos iguais (intervalo de tempo do MCPE).
def equations(t, y): #Define a função que depende dos parâmetros "t" e "y" em que "t"  é o tempo e "y" é a população e retorna a derivada de "y" em "t".
    dydt = k1*y  #Equação correspondente a derivada de "y" com relação a "t".
    return np.array([dydt], float) #Retorna um vetor com a derivada de "y" com relação a "t".
y0 = [50] #Condição inicial para o modelo MCPE (tamanho inicial da população).
solution = scp.solve_ivp(equations, [0, time[-1]], y0, t_eval=time) #Calcula a solução numérica da equação dydt com valor inicial y0=50 no intervalo de tempo [0, 50].
fig = plt.figure(figsize=( 5, 5)) #Define as dimensões do gráfico em 5 polegadas cada.
plt.plot(time,solution.y[0],"blue") #Imprime em cor azul a solução do MCPE no intervalo [0,50].
blue_patch = mp.Patch(color="blue", label="População") #Constrói legenda.
```

```
plt.legend(handles=[blue_patch]) #Adiciona o quadro de legenda no gráfico.
plt.title("Modelo Populacional Exponencial") #Adiciona o título no gráfico.
plt.xlabel("Tempo") #Nomeia o eixo x.
plt.ylabel("Populacão") #Nomeia o eixo y.
plt.show() #Imprime o gráfico.
```

Modelo 4: Modelo epidemiológico com indivíduos suscetíveis, infectados e recuperados (SEDO)

```
import numpy as np #Importa a biblioteca numpy.
import matplotlib.pyplot as plt #Importa o módulo pyplot da biblioteca matplotlib.
import matplotlib.patches as mp #Importa o módulo patches da biblioteca matplotlib.
import scipy.integrate as scp #Importa o módulo integrate da biblioteca scipy.
alpha = 0.005 #Parâmetro alpha das equações que descrevem o modelo SIR
beta = 0.2 #Parâmetro beta das equações que descrevem o modelo SIR
time = np.linspace(0,80,1000) #Cria um vetor com uma partição do intervalo [0,80] com
1000 pedaços de tamanhos iguais (intervalo de tempo do modelo SIR).
def equations(t, y): #Define função na qual  "y" é o vetor cujo as entradas correspondem
às populações dos grupos "S", "I" e "R", respectivamente,  e "t" é variável  associada ao
tempo. Esta função retorna  um vetor com as derivadas no instante de tempo "t".
    S,I,R = y #Escreve nas variáveis "S", "I" e "R" os valores atuais de "y".
    dSdt = -alpha*S*I #Derivada de "S" com relação a  "t".
    dIdt = alpha*S*I - beta*I  #Derivada de "I" com relação a "t".
    dRdt = beta*I  #Derivada de "R" com relação a "t".
    return np.array([dSdt,dIdt,dRdt], float) #Retorna um vetor com as derivadas de "S", "I"
e "R" com relação a "t".
y0 = np.array([99,1,0], float) #Condições iniciais para o modelo SIR (99% da população
suscetível, 1% infectada e 0% da recuperada).
solution = scp.solve_ivp(equations, [0,time[-1]], y0, t_eval = time) #Calcula a solução
numérica do sistema de equações  no intervalo [0, 80] com as condições iniciais
mencionadas anteriormente.
fig = plt.figure(figsize=( 5, 5)) #Define as dimensões do gráfico em 5 polegadas cada.
black_patch = mp.Patch(color="black", label="Suscetíveis") #Constrói legenda em cor
preta para "S(t)".
red_patch = mp.Patch(color="red", label="Infectados") #Constrói legenda em cor
vermelha para "I(t)".
```

```
blue_patch = mp.Patch(color="blue", label="Recuperados") #Constrói legenda em cor azul para "R(t)".
plt.legend(handles=[black_patch,red_patch,blue_patch]) #Adiciona legendas no gráfico.
plt.title("Modelo SIR") #Adiciona o título no gráfico.
plt.xlabel("Tempo")  #Nomeia o eixo x.
plt.ylabel("População") #Nomeia o eixo y.
plt.plot(time,solution.y[0], color="black") #Imprime "S(t)" em cor preta no intervalo [0,80].
plt.plot(time,solution.y[1], color="red") #Imprime "I(t)" em cor vermelha no intervalo [0,80].
plt.plot(time,solution.y[2],color="blue")  #Imprime "R(t)" em cor azul no intervalo [0,80].
plt.show()  #Imprime o gráfico.
```

MODELAGEM MATEMÁTICA E *PYTHON*: FERRAMENTAS PARA O ENSINO-APRENDIZADO DE MATEMÁTICA

Na perspectiva da prática como parte fundamental para a consolidação dos seus conhecimentos e como prática da teoria sociointeracionista entre professor e aluno assim como defendido por Junior e Onuchic (2015), deixaremos algumas sugestões de atividades das quais necessitam de conhecimento sobre aritmética dos inteiros, cálculo numérico e cálculo diferencial.

Atividade 1. Uma *sequência de Fibonacci* é uma construção numérica recorrente no qual $F_1 = 1$, $F_2 = 1$ e os demais elementos regidos pela regra $F_n = F_{n-1} + F_{n-2}, n \geq 3$. Gere os 100 primeiros elementos desta sequência utilizando a linguagem de programação *Python* no ambiente *Google Colab*.

Atividade 2. Uma *progressão geométrica* é uma sequência numérica denotada por (a_n)tal que a_1é escolhido e, para todo n, número natural, tem-se que $a_{n+1} = a_n . q$, no qual q é um número real fixo, diferente de 0 e de 1, chamado razão. Mostre que $a_n = a_1 . q^{n-1}$e gere os 100 primeiros elementos desta sequência utilizando a linguagem de programação *Python* no ambiente *Google Colab*. Como diversão, constate que a soma dos *n* primeiros elementos da sequência é dada pela fórmula $S_n = a_1 . \frac{q^n - 1}{q - 1}$.

Atividade 3. A tabela 2 fornece dados sobre comprimento e peso de Tilápias do rio Nilo. Gere um código utilizando a linguagem de programação *Python* no ambiente *Google Colab* que forneça um modelo de regressão não linear aplicando o Método dos Mínimos Quadrados. Para isso, sugerimos que o leitor utilize uma função do tipo $\varphi(x,\beta) = \beta_1 x^{\beta_2}$ e realize o processo de linearização antes de aplicar o MMQ.

Tabela 2: Dados sobre comprimento e peso de Tilápias do rio Nilo.

x: comprimento	11	15	17.4	20.6	22.7	25.3	27.4	28.2	29.3
f(x): peso	26	59.5	105.4	202.2	239.5	361.2	419.8	475.4	488.2

Fonte: Bassanezi (2002).

Atividade 4. O MSEIR é um *modelo compartimental epidemiológico não linear*. Nesse modelo as pessoas são divididas em 5 grupos, *Imunes* (M), *Suscetíveis* (S), em estado de *Latência* (E), *Infectados* (I) e *Removidos* (R). O modelo supõe que o indivíduo nasce com uma imunidade temporária e depois de determinado tempo passa para o grupo de Suscetíveis. Da mesma forma, uma pessoa que está suscetível e teve contato com infectado passa para o grupo de estado de Latência. Caso esse indivíduo desenvolva a doença, então ele passa para o grupo de Infectados e em seguida para o grupo de Removidos depois de um período infeccioso. Além disso, é considerado que um indivíduo pode vir a óbito em qualquer um desses grupos por outros motivos além da doença. A formulação deste modelo também se dá através de um SEDO não linear, como podemos observar:

$$\begin{cases} \dfrac{\mathrm{d}M}{\mathrm{d}t} = \mu(N-S) - (\delta+\mu)M, \\ \dfrac{\mathrm{d}S}{\mathrm{d}t} = \mu S + \delta M - \beta SI - \mu S, \\ \dfrac{\mathrm{d}E}{\mathrm{d}t} = \beta SI - (\varepsilon+\mu)E, \\ \dfrac{\mathrm{d}I}{\mathrm{d}t} = \varepsilon E - (\gamma+\mu)I, \\ \dfrac{\mathrm{d}R}{\mathrm{d}t} = \gamma I - \mu R \end{cases}$$

em que, N representa a população total dada por $S(t) + M(t) + E(t) + I(t) + R(t)$, δ a taxa de imunidade temporária, μ a taxa média de morte, $\frac{1}{\varepsilon}$ o período de latência, $\frac{1}{\gamma}$ o período infeccioso médio, β a taxa de contágio e SI representa o encontro entre suscetível e infectado. Gere a simulação desse modelo utilizando a linguagem de programação *Python* no ambiente *Google Colab*, considerando as taxas $\mu = 0.03, \delta = 0.5, \frac{1}{\varepsilon} = 0.4, \frac{1}{\gamma} = 0.4$ e $\beta = 0.6$.

Informamos que o leitor encontrará parte da Atividade 3 desenvolvida em Silva (2021) e os códigos estão disponíveis na plataforma *GitHub* por meio do *link* ModelosMatematicos/ModelosMatematicosPython (github.com).

CONSIDERAÇÕES FINAIS

Neste trabalho, nosso objetivo é apresentar a construção de modelos matemáticos utilizando a linguagem de programação *Python* como motivador para o estudo do cálculo diferencial e lógica matemática, tendo em vista o índice de reprovação e evasão no curso de Licenciatura Plena em Matemática da UFRPE.

O conteúdo deste estudo deriva do minicurso intitulado "*Python* e Aplicações Matemáticas", que está vinculado ao projeto de ensino denominado "Ciclo de Atividades Complementares do Curso de Licenciatura em Matemática da UFRPE", bem como de uma monografia elaborada como parte do curso de Licenciatura em Matemática sendo uma proposta alternativa de ensino-aprendizagem das disciplinas de Cálculo Diferencial e Lógica Matemática.

Reconhecemos que as técnicas exploradas neste estudo evidenciam a importância da Modelagem Matemática no ensino e aprendizado do cálculo diferencial, permitindo aos estudantes se envolverem com a metodologia científica. Os modelos de dinâmica populacional e epidemiológica descritos fornecem uma breve exposição à matemática empregada na construção dos modelos amplamente divulgados pela mídia para monitorar a evolução da pandemia da COVID-19.

Identificamos que o domínio de conceitos de lógica matemática, como conectivos, condicionais, tabelas-verdade e técnicas de demonstrações, facilita a construção de códigos em diversas linguagens de programação. Escolhemos *Python* devido à sua curva de aprendizado rápida e à sua popularidade entre cientistas, analistas e pesquisadores. Essa

escolha é justificada pela ampla disponibilidade de bibliotecas, como Matplotlib, NumPy, SymPy, Pandas e outras, especialmente projetadas para o desenvolvimento de algoritmos, visualização, coleta e análise de dados.

Ressaltamos que foi utilizado o ambiente *Google Colaboratory* devido à sua acessibilidade e facilidade de execução dos códigos em *Python*. Além de dispensar a necessidade de configuração ou instalação de *softwares* no computador do leitor.

Em nossa pesquisa, não encontramos educadores que sejam explicitamente contra o estudo de cálculo diferencial e lógica matemática com *Python*. De forma geral, a comunidade educacional e acadêmica reconhece o valor do uso de linguagens de programação, como *Python*, para auxiliar na aprendizagem desses temas. No entanto, é importante lembrar que diferentes educadores podem ter abordagens diversas em relação ao uso de linguagens de programação no ensino. Algumas escolas ou educadores podem preferir métodos tradicionais no ensino do cálculo diferencial e lógica matemática, sem o uso de linguagens de programação. Essa preferência pode ser baseada em fatores como currículo escolar, recursos disponíveis ou experiências profissionais. Além disso, é importante reconhecer que o *Python* oferece uma ferramenta valiosa para a aplicação prática e visualização dos conceitos matemáticos, enriquecendo a experiência de aprendizado.

As atividades sugeridas possibilitam aos estudantes, interessados por Matemática Aplicada, compreender, explorar e consolidar conceitos matemáticos por meio do contato com a programação, proporcionando a construção do conhecimento de forma ativa. Entendemos que essa prática pode contribuir não apenas para o aprendizado da matemática, mas também para estimular a participação dos estudantes em pesquisas científicas.

Finalmente, gostaríamos de enfatizar que estamos à disposição para esclarecer dúvidas sobre o assunto, trocar de informações, propor novos estudos, entre outros.

REFERÊNCIAS

BASSANEZI, Rodney Carlos. **Ensino-aprendizagem com modelagem matemática:** uma nova estratégia. 4. ed. São Paulo: Contexto, 2002.

BOYCE, Wiliam; DIPRIMA, Richard. **Equações Diferenciais Elementares e Problemas de Valor de Contorno**. 8. ed. Rio de Janeiro: LTC Editora, 2010.

GONÇALVES, Mirian Buss; FLEMMING, Diva Marília. **Cálculo B:** Funções de várias variáveis, integrais múltiplas, integrais curvilíneas e de superfícies. 2. ed. São Paulo: Pearson, 2007.

GUIDO VAN ROSSUM. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2021. Disponível em: https://pt.wikipedia.org/w/index.php?title=Guido_van_Rossum&oldid=61161128. Acesso em: 16 mai. 2021.

JUNIOR, Luiz Carlos Leal; ONUCHIC, Lourdes de la Rosa. Ensino e Aprendizagem de Matemática Através da Resolução de Problemas Como Prática Sociointeracionista. **BOLEMA**: Boletim de Educação Matemática, Rio Claro, v. 29, p. 955-978, 2015. 1980-4415. DOI: https://doi.org/http://dx.doi.org/10.1590/1980-4415v29n53a09. Disponível em: https://www.scielo.br/j/bolema/a/nLsFMY58vc7767N6RV9rGcb/?lang=pt. Acesso em: 13 abr. 2020.

MACÊDO, José Antunes de; MADUREIRA, Romário Santos; CASTRO, Adriana Martins da Silva. Modelagem matemática aplicada a evapotranspiração de referência para a cidade de Januária (MG). **Revista Sergipana de Matemática e Educação Matemática**, Sergipe, 18 set. 2020. Semestral. Disponível em: https://doi.org/10.34179/revisem.v5i2.13420. Acesso em: 16 ago. 2021.

PRADO, Maria Elisabette Brisola Brito *et al*. Pensamento computacional e atividade de programação: perspectivas para o ensino da matemática. **Revista Sergipana de Matemática e Educação Matemática**, Sergipe, 18 set. 2020. Semestral. Disponível em: https://doi.org/10.34179/revisem.v5i2.14422. Acesso em: 16 ago. 2021.

RAMOS, Luiz Fernando Pinheiro; FERNANDES, Humberto Cesar; BATISTA, Ben Dêivide de Oliveira. Modelagem Matemática para Previsão de Jogos de Futebol. **Revista Sergipana de Matemática e Educação Matemática**, Sergipe, 26 jun. 2021. Semestral. Disponível em: https://doi.org/10.34179/revisem.v6i1.13637. Acesso em: 16 ago. 2021.

ROCHA, Ana Karina de Oliveira; PRADO, Maria Elisabette Brisola Brito; VALENTE, José Armando. A linguagem de programação Scratch na formação do professor: uma abordagem baseada no TPACK. **Revista Sergipana de Matemática e Educação Matemática**, Sergipe, 18 set. 2020. Semestral. Disponível em: https://doi.org/10.34179/revisem.v5i2.14421. Acesso em: 16 ago. 2021.

RUGGIERO, Marcia A. Gomes; LOPES, Vera Lúcia da Rocha. **Cálculo Numérico:** Aspectos Teóricos e Computacionais. 2. ed. São Paulo: Pearson, 2000.

SCIPY DEVELOPERS. **Numpy and Scipy Documentation.** *[S.l.].* SciPy developers, 2021. Disponível em: https://docs.scipy.org/doc/. Acesso em: 4 set. 2021.

SILVA, Angelo Antunes da Rocha. **Técnicas de Modelagem Matemática e os Métodos de Runge-Kutta.** Orientador: Maria Ângela Caldas Didier. 2021. 90 f. TCC (Graduação) - Curso de Licenciatura em Matemática, DM, Universidade Federal Rural de Pernambuco, Recife, 2021. Disponível em: http://hdl.handle.net/123456789/2710. acesso em: 4 set. 2021.

STEWART, James. **Cálculo.** 8. ed. São Paulo: Cengage Learning, v. 1, 2017.

SYMPY DEVELOPMENT TEAM. **Welcome to SymPy's documentation!.** *[S.l.].* SymPy Development Team, 2021. Disponível em: https://docs.sympy.org/latest/index.html. Acesso em: 4 set. 2021.

THE MATPLOTLIB DEVELOPMENT TEAM. **Overview.** *[S.l.].* The Matplotlib development team, 2021. Disponível em: https://matplotlib.org/stable/contents.html. Acesso em: 4 set. 2021.

THE NUMPY COMMUNITY. **NumPy v1.21 Manual.** *[S.l.].* The NumPy community, 2021. Disponível em: https://numpy.org/doc/stable/. Acesso em: 4 set. 2021.

TANG, B. et al. (2021). Modeling the COVID-19 epidemic in the United States using a compartmental model. Mathematical Biosciences and Engineering, 18(1), 1331-1352.

NORVIG, P. (1992). Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp. Morgan Kaufmann Publishers.

Submetido em 04 de novembro de 2022.
Aprovado em 04 de dezembro de 2023.