RESOLUÇÃO DE EDO'S UTILIZANDO MÉTODOS NUMÉRICOS DE ORDEM SUPERIOR COM LINGUAGEM PYTHON

EDO SOLUTIONS USING HIGHER ORDER NUMERICAL METHODS WITH PYTHON LANGUAGE

Tiago Destéffani Admiral

Instituto Federal de Educação e Tecnologia Fluminense - IFF, Campos dos Goytacazes - RJ. E-mail: tdesteffani@gmail.com

Este artigo tem o objetivo de descrever, por meio de dois exemplos, como podemos utilizar a plataforma Python para implementar um código de obtenção de solução numérica para problemas de valor inicial, EDO's com condições de contorno. Um dos exemplos é a solução de uma EDO que modela o movimento de uma máquina de Atwood para uma das massas variando com o tempo, o outro é o exemplo de uma barra em queda livre sendo, simultaneamente, desacelerada por uma força magnética variável. A metodologia do trabalho se baseou em utilizar o método de Runge Kutta de quarta ordem para obtenção dos autovalores dentro de um intervalo de tempo e, posteriormente, comparar os autovalores com os valores exatos da função solução analítica das EDO's. Os resultados mostraram concordância com desvio padrão médio da ordem de $\sigma = 10^{-6}$ além de ter se mostrado, com poucas adaptações, uma opção para soluções de diversos PVI's, comuns na física.
Palavras chave: Ensino de física, EDO, PVI, Python

This article aims to describe, through two examples, how we can use the Python platform to implement a code to obtain a numerical solution for initial value problems, ODE's with boundary conditions. One example is the solution of an ODE that models the motion of an Atwood machine for one of the masses varying with time, the other is the example of a bar in free fall being simultaneously decelerated by a variable magnetic force. The methodology of the work was based on using the fourth order Runge Kutta method to obtain the eigenvalues within a time interval and, later, comparing the eigenvalues with the exact values of the analytical solution function of the ODE's. The results showed agreement with an average standard deviation of the order of $\sigma = 10^{-6}$ in addition to having shown, with few adaptations, an option for solutions of several PVI's, common in physics.
Keywords: Physics teaching, ODE, PIV, Python

INTRODUÇÃO

Durante o processo de modelagem matemática de um fenômeno físico é muito comum que o problema matemático gerado seja uma Equação Diferencial Ordinária (EDO) ou uma Equação Diferencial Parcial (EDP). Exatamente por essa razão é importante que, durante sua formação o aluno aprenda a obter a solução analítica desses tipos de equações (BOYCE, 2020; KRANTS, 2005).

Entretanto, nem toda EDO possui uma solução analítica. Para sermos mais precisos podemos dizer que apenas algumas EDO possuem solução analítica. Mesmo assim, embora alguns métodos numéricos sejam ensinados nos cursos de física, de maneira geral, existe pouca ênfase (CARUSO e OGURI, 2014) na utilização desses métodos, durante a formação docente. Entretanto literatura é crescente a aparição do tema em trabalhos relacionados como (CARUSO e OGURI, 2022; MENDONÇA, 2022) que utilizam a linguagem Python para aproximações numéricas, entretanto utilizando métodos distintos.

Esse artigo descreve a obtenção da solução numérica de dois exemplos de problemas de (PVI), utilizando o Python, trazendo uma comparação entre os autovalores com os valores exatos da solução analítica. Diferente de outros trabalhos encontrados na literatura (CARUSO e OGURI, 2022; MENDONÇA, 2022; QUINGA, 2021), usaremos um método simples e de boa precisão, o método clássico de passo único, Runge Kutta de quarta ordem. Esse método numérico é, provavelmente, um dos mais utilizados para solução de PVI's, pelo fato de ser capaz de fornecer resultados extremamente

precisos sem a necessidade de que o passo (h) seja muito pequeno, o que o torna de fácil implementação computacional, e também o faz requerer pouca capacidade de processamento (JUSTO, et. al., 2020).

MÉTODOS NUMÉRICOS

De maneira geral os métodos numéricos são chamados de iterativos por precisarem de uma repetição de operações específicas, para a obtenção da solução. Os métodos numéricos iterativos para solução de PVI podem ser classificados de como métodos com memória ou sem memória (CHICHARRO, 2017). Os métodos sem memória sendo aqueles em que a próxima iteração depende apenas do valor da iteração imediatamente anterior, ao passo que, nos métodos com memória, uma iteração pode depender do valor de várias iterações anteriores.

Entre os métodos numéricos utilizados para obter a solução de um PVI, os mais utilizados são os métodos de Runge Kutta, que podem ser classificados como sem memória. Esses métodos são generalizações do método de Euler melhorado, em que são inseridos mais estágios de cálculos, obtendo-se ordens de precisão mais altas (JUSTO, 2020). Desenvolvido pelo matemático alemão Carl Runge, e posteriormente aperfeiçoado pelo matemático, também alemão Martin Kutta, o método clássico de Runge Kutta de quarta ordem (RK4) (ou de quatro estágios) pode fornecer valores com erro relativo inferior a 0,0002% (KRANTS, 2005), se tomado com passo 0,1.

Dado um PVI, envolvendo uma EDO de primeira ordem, tal que (JUSTO, 2020):

$$\frac{dy}{dx} = f(x,y) \quad e \quad y(x_o) = y_o \tag{1}$$

A aproximação do termo y_{n+1}, pelo método RK4, é dado pela equação 2:

$$y(x_{n+1}) = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{2}$$

Em que;

$$x_{n+1} = x_n + h \tag{3}$$

E a quantidade h é chamada de passo, quanto menor é o seu valor, para um mesmo número de iterações realizadas, maior é a precisão do método. Os coeficientes k_n são dados pelas equações:

$$k_1 = f(x_n, y_n) \tag{4}$$

$$k_2 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \tag{5}$$

$$k_3 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2) \tag{6}$$

$$k_4 = f(x_n + h, y_n + hk_3) \tag{7}$$

Através das equações de iteração (eq.2 a eq.6) percebemos que o autovalor relacionado a y_{n+1} depende dos valores das constantes k_n, porém, em relação aos termos iterados, depende exclusivamente do termo anterior y_n.

Outra vantagem do método RK4 é que, diferente de outros métodos, não é necessário conhecer nenhum valor de $\frac{dy}{dx}$, uma vez que a aproximação leva em consideração a determinação dos coeficientes k_n (JUSTO, 2020).

METODOLOGIA

A seguir vamos apresentar dois exemplos de problemas de valor inicial com suas soluções analíticas e suas soluções numéricas.

Exemplo 1 – Problema de Atwood com massa variável

A situação do primeiro exemplo consiste em dois objetos, de massas m_1 e m_2, acoplados por um fio inextensível e de massa desprezível, conectados por meio de uma polia também de massa desprezível, o problema foi retirado de (HALLIDAY, 2009), a situação é ilustrada na Figura a seguir:

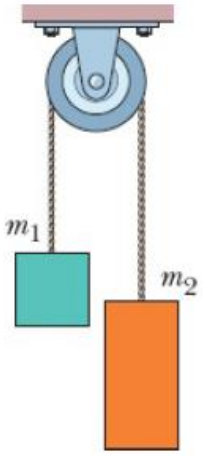

Figura 1: Máquina de Atwood (HALLIDAY, 2009).

Na situação descrita, ao se liberar o movimento, consideramos que o corpo m_1 perde massa à uma taxa constante k (kg/s), de forma que sua massa varia com o tempo. Pelo fato de a massa variar com o tempo, não podemos utilizar a segunda Lei de Newton em sua forma "convencional", dessa forma utilizaremos a prerrogativa que a força resultante equivale à taxa de variação do momento linear total, conforme a equação (8)

$$\overrightarrow{F_r} = \frac{d\vec{p}(t)}{dt} \qquad (8)$$

Sabemos que m_1 perde massa a uma taxa constante, logo podemos escrever: $m_1(t) = m_{10} - kt$, em que m_{10} é o valor da massa para $t = 0s$. Tomando-se o momento linear total do sistema, considerando a força líquida atuante sobre este, podemos reescrever a equação (8):

$$(m_2 - m_{10} + kt)g = \frac{d[(m_2+m_{10}-kt).v(t)]}{dt} \qquad (9)$$

Utilizando a regra do produto no membro da direita e reorganizando a equação (9), teremos que:

$$\frac{dv(t)}{dt} - \frac{k}{(m_2+m_{10}-kt)}v(t) = \frac{(m_2-m_{10}+kt).g}{(m_2+m_{10}-kt)} \qquad (10)$$

A equação (10) é uma EDO de primeira ordem, dessa forma podemos utilizar o método do fator integrante (BOYCE, 2020) para obter a função $v(t)$. O fator integrante é determinado através da expressão:

$$e^{\int \frac{-k}{(m_2+m_{10}-kt)}dt} = m_2 + m_{10} - kt \qquad (11)$$

Multiplicando a equação (10) pela equação (11), o primeiro membro pode ser escrito como uma regra do produto entre $[v(t)]$ e $[m_2 + m_{10} - kt\,]$, integrando ambos os membros em relação à t, e isolando para velocidade teremos:

$$v(t) = \frac{g}{m_2+m_{10}-kt}(m_2 t - m_{10}t + \frac{k}{2}t^2 + C) \qquad (12)$$

Utilizando como condição inicial que o sistema parte do repouso, teremos que $v(0) = 0$, aplicando essa condição de contorno à equação (12), concluímos que a constante C é nula, e simplificando teremos a equação (13):

$$v(t) = \frac{gt(\frac{k}{2}t+m_2-m_{10})}{m_2+m_{10}-kt} \qquad (13)$$

A equação (13) é a solução analítica para o PVI regido pela equação (10), uma equação geral que pode ser utilizada para obter a função da velocidade para qualquer combinação de parâmetros: $m_{10}, m_2, g\ e\ k$.

Solução numérica

Utilizando o software livre Spyder (Python 3.9), implementamos o método de Runge Kutta de quarta ordem para obter os valores da função $v(t)$. Como metodologia de validação foi realizada uma comparação entre os valores obtidos diretamente na solução analítica com os autovalores obtidos numericamente. Os parâmetros utilizados foram:

$m_{10} = 2{,}00\ kg$

$m_2 = 4{,}00\ kg$

$g = 9{,}80\ m/s^2$

$k = 0{,}15\ kg/s$

O intervalo observado foi $[0, 20]$ segundos, tendo sido realizadas 200 iterações, dessa forma o passo h, para o método de Runge Kutta, ficou definido como 0,1. O código em Python para essa EDO se encontra no Apêndice. O algoritmo tem como base todas as equações de (2) a (6), sendo construído com base na função a lógica "for", tendo sua limitação no número (N) de iterações que pode ser ajustada de acordo com a precisão requerida no problema (SHAMPINE, 1986).

O gráfico mostra os autovalores da função $v(t)$ obtidos numericamente pelo Python:

Gráfico 1: Solução numérica da função $v(t)$.

O gráfico 1 foi gerado automaticamente pelo próprio Python, com o auxílio da biblioteca *matplotlib.pyplot*, a biblioteca possui diversas funcionalidades para inclusão de informações no gráfico,

bem como alterações nas formas de exibição. Em termos da comparação entre os autovalores obtidos com os valores exatos da função analítica, obtivemos desvio padrão médio de $\sigma = 1{,}0.10^{-6}$. Alguns valores são mostrados na Tabela 1:

$V(t)\ x\ t$	*0,1*	*0,2*	*0,3*	*0,4*	*0,5*
Solução exata	0,328713	0,661541	0,998514	1,339663	1,685021
Solução numérica	0,328713	0,661541	0,998514	1,339662	1,6785021
$V(t)\ x\ t$	**0,6**	**0,7**	**0,8**	**0,9**	**1,0**
Solução exata	2,034619	2,388490	2,746667	3,109182	3,476068
Solução numérica	2,034618	2,388489	2,746667	3,109182	3,476068

Tabela 1: Comparação de valores obtidos numericamente e solução exata.

Observe que os valores da Tabela 1 são mostrados com seis casas após a vírgula, pois utilizando a função de arredondamento até a quinta casa não havia diferença perceptível nos valores. Só após a sexta casa ser considerada os dados mostraram eventual discordância e, como afirmado em seções anteriores, essa precisão pode ser considerada razoável, para propósitos didáticos.

Podemos também utilizar o programa para simular outros cenários como, por exemplo, o caso em que $m_{10} > m_2$, ou ainda casos em que há uma velocidade inicial do sistema, podendo essa última ser ou não na direção da força resultante. Para realizar essas modificações basta alterar os parâmetros do programa.

Ao fazermos, por exemplo, a modificação $m_{10} = 5{,}00\ kg$, teremos inicialmente um movimento acelerado no sentido oposto ao verificado no primeiro exemplo, porém, conforme o objeto m_1 vai tendo sua massa reduzida a aceleração vai sendo reduzida, até eventualmente se anular e posteriormente, mudar de direção.

Modificamos o parâmetro m_1 juntamente com o range das iterações, mantendo o passo $h = 0{,}1$. O gráfico 2 mostra, à esquerda, os valores da solução exata obtida pela solução analítica, e à direita os autovalores obtidos numericamente.

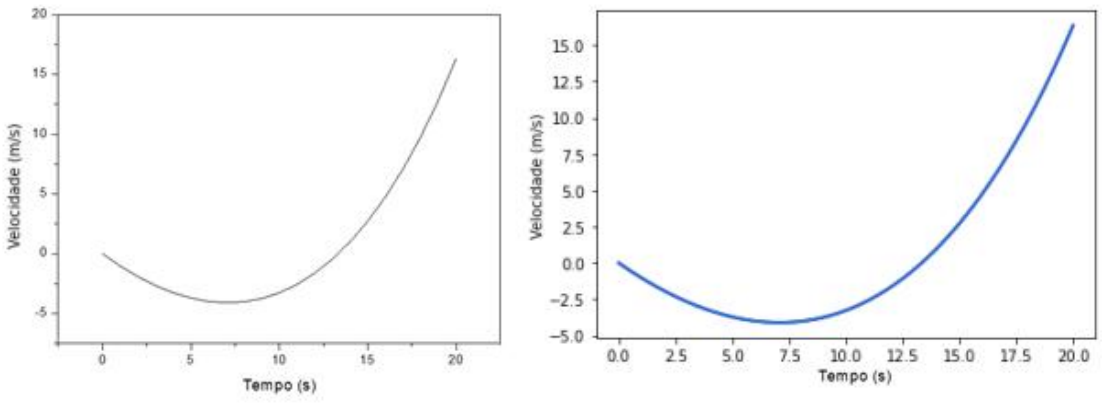

Gráfico 2: Comparação gráfica das soluções numéricas e exatas.

Como o desvio padrão entre os dados é muito pequeno se tornou inviável colocar as duas curvas no mesmo plano cartesiano, pois estas se sobrepõem, não sendo possível uma visualização gráfica da diferença entre elas.

Pode ser visto no Gráfico 2 que foi tomado um intervalo de tempo, devido ao fato de que para essa situação o sistema leva um tempo maior para convergir para o regime de movimento previsto. Observou-se que a precisão do método, para essas novas condições iniciais, permaneceu o mesmo, como esperado.

Exemplo 2 – Barra caindo e sujeita à uma força elétrica

O segundo exemplo é um problema em que uma barra metálica está acoplada, de forma que pode deslizar com atrito desprezível, em um trilho em forma de "U", também metálico, que possui uma resistência elétrica R. Toda a região interna do trilho é permeada por um campo magnético $\vec{B}$ que aponta para dentro do plano da página, e a barra está sujeita à uma aceleração da gravidade $\vec{g}$. O problema foi adaptado de (ADMIRAL, 2016) em que foi abordado de forma apenas analítica. A situação é mostrada na Figura:

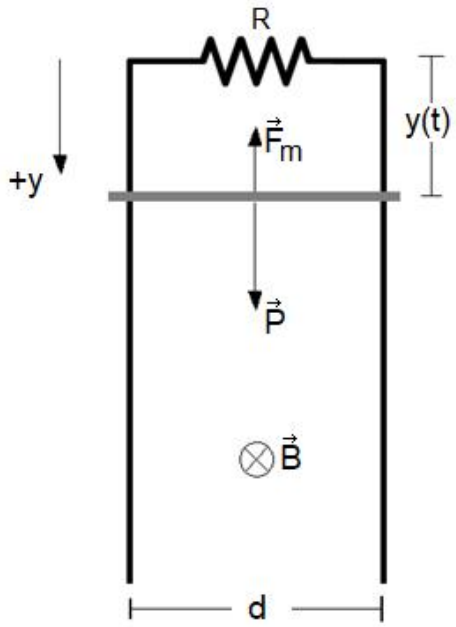

Figura 2: Situação do segundo exemplo.

Analisando as forças que atuam sobre a barra metálica na direção $\hat{y}$ durante seu movimento podemos escrever a equação (14):

$$m\vec{g} - \vec{B}dI(t) = m\vec{a}(t) \qquad (14)$$

Em que m é a massa da barra, em kg, d é o comprimento da barra, em metros, I(t) e a(t) representam os valores de corrente elétrica, em ampère, e aceleração da barra, em m/s, com o tempo, respectivamente. Conforme a barra se desloca ocorrerá uma variação na área permeada pelo campo magnético com o tempo $(A(t))$. Isso acarretará numa variação do fluxo magnético através do circuito fechado estabelecido entre a barra e os trilhos, tal variação de fluxo magnético gera, de acordo com a Lei de Faraday, uma força eletromotriz ε induzida (GRIFFITHS, 2011) dada pela equação (15):

$$\varepsilon = -\frac{d\Phi}{dt} \qquad (15)$$

Em que Φ é o fluxo magnético, dado pela equação (16) (GRIFFITHS, 2011):

$$\Phi = \oint_s \vec{B} \cdot d\vec{A} \qquad (16)$$

Para esse caso, de maneira particular, temos que o campo magnético $\vec{B}$ é constante. A direção do vetor $d\vec{A}$ é dada pelo vetor normal $\hat{n}$ que aponta para fora do plano da página, dessa forma sabemos que o ângulo entre os vetores $\vec{B}$ e $d\vec{A}$ vale π. Usando esses parâmetros na equação (16), e substituindo o resultado na equação (15) teremos, em módulo, que:

$$\varepsilon = -\frac{d[BA(t).\cos(\pi)]}{dt} = Bd\frac{d[y(t)]}{dt} \qquad (17)$$

Utilizando o resultado da força eletromotriz da equação (17) na Lei de Ohm podemos escrever a função que descreve o módulo da corrente elétrica $I(t)$, considerando a resistência elétrica R, como:

$$I(t) = \frac{B.d}{R}v(t) \qquad (18)$$

Considerando que $\frac{d[y(t)]}{dt}$ representa a velocidade instantânea $v(t)$ com que a barra se move, sabendo que a aceleração vale $a(t) = \frac{d[v(t)]}{dt}$, e utilizando a relação encontrada em (18), podemos reescrever a EDO da equação (14) como:

$$\frac{d[v(t)]}{dt} + \left(\frac{B^2d^2}{R.m}\right)v(t) - g = 0 \qquad (19)$$

A equação (19) é uma EDO de primeira ordem e pode ser resolvida analiticamente pelo método do fator integrante (KRANTS, 2005). Utilizando a condição de contorno $v(0) = 0m/s$ na solução geral da EDO, teremos a solução particular do PVI:

$$\vec{v}(t) = \frac{R.m.g}{B^2d^2}\left[1 - e^{-\left(\frac{B^2d^2}{R.m}\right)t}\right]\hat{y} \qquad (20)$$

A equação (20) apresenta uma característica típica de sistemas que convergem para um valor específico de equilíbrio (KRANTS, 2005), nesse caso é a velocidade limite (ou velocidade terminal) que pode ser obtida tomando-se o limite para valores de tempo muito grandes, que resulta em:

$$\lim_{t\to\infty} v(t) = \left(\frac{R.m.g}{B^2d^2}\right) \qquad (21)$$

A equação (21) nos fornece o valor da velocidade limite para quaisquer combinações de valores dos parâmetros $R, m, B, g\ e\ d$.

Solução numérica

Utilizando a mesma metodologia aplicada anteriormente foi implementado um código em Python para determinação da solução numérica da equação (19), por meio do método de Runge Kutta de quarta ordem, no intervalo $[0, 20]$ segundos, os dados foram comparados com os valores da função analítica para o mesmo intervalo de tempo. Os parâmetros utilizados foram:

$B = 750\ mT$

$g = 9{,}81\ m/s^2$

$m = 50\ g$

$d = 0{,}30\ m$

$r = 4{,}70\ \Omega$

Foram realizadas 200 iterações, com passo $h = 0.1$. O número de iterações se mostrou suficiente para gerar uma aproximação com baixo erro absoluto. A estrutura do código, que se encontra no apêndice, é similar ao utilizado no primeiro problema, uma vez que foi utilizado o mesmo método numérico. O gráfico mostra a curva da variação da velocidade em função do tempo:

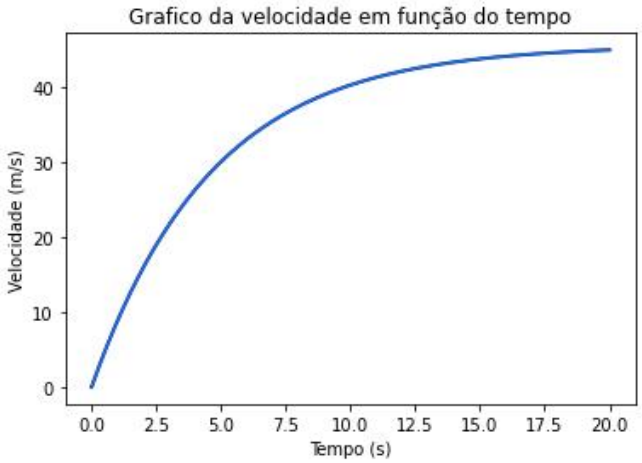

Gráfico 3: Solução numérica de $v(t)$.

Diferentemente do primeiro exemplo, podemos ver que a velocidade tende a um valor com o passar do tempo, esse valor é a velocidade limite que, de acordo com a previsão teórica (equação 21), para os valores de parâmetros selecionados possui o valor aproximado de $45{,}538 m/s$.

Tanto o intervalo de tempo, quanto o valor do passo de iteração, estipulados foram suficientes para perceber a convergência do valor esperado. No gráfico 3 são mostrados os resultados no intervalo de [0 ,20s], ao fim desse intervalo o valor da velocidade obtida corresponde 98,7% do valor limite.

Outra análise que podemos fazer diz respeito a constante de tempo, sabemos que, para sistemas que convergem para um valor estabilizado, a constante de tempo é o valor de tempo necessário para que se atinja 63,2% do valor final (OGATA, 2003). Tal constante, obtida pelo inverso da expressão que aparece no termo do expoente da equação (20), tem seu valor esperado de, aproximadamente, 4,6 para os parâmetros escolhidos. Durante a simulação constatamos uma correspondência com o valor obtido e o valor esperado.

CONSIDERAÇÕES FINAIS

A utilização do Python para resolver EDO's numericamente se apresentou como uma alternativa viável para conhecer, com boa precisão, e de maneira rápida a solução de PVI's, e o método utilizado, embora de passo único, foi capaz de prover soluções com boa precisão para propósitos didáticos.

Percebeu-se que o Python possui uma estrutura de programação amigável e que, com poucas adaptações para a mesma estrutura de algoritmo, podemos solucionar um número muito grande de PVI's. Esse resultado é particularmente relevante, dado o número grande de possíveis aplicações para o método, que pode ser aplicado de maneira análoga à sistema de EDO's, para solução de EDO's de ordem superior.

Entre as sugestões que podem se apresentar para a utilização nas aulas de física, podemos mencionar a facilidade de que existem compiladores Python online, ou seja, o professor pode utilizar esse recurso durante as aulas para que os alunos obtenham uma solução de um PVI, seja para exemplificar uma explicação, ou para comparar um resultado obtido analiticamente.

Essa segunda aplicação se torna particularmente útil, uma vez que os parâmetros do PVI podem ser completamente modificados, gerando soluções distintas a cada modificação, o que pode ser um exercício de previsão dos resultados esperados, algo similar ao que foi abordado no exemplo 1.

REFERÊNCIAS

ADMIRAL, T. D. Dificuldades conceituais e matemáticas apresentadas por alunos de física dos períodos finais. **Revista Brasileira de Ensino de Física**, vol. 38, nº 2, e2502, 2016.

BOYCE, E. **Equações Diferenciais Elementares e Problemas de Valores de Contorno**, São Paulo, 11ª Edição, 2020.

Krants. S **Differential Equations Demystified**. McGRAW-HILL, 2005.

CARUSO, F. OGURI, V. O método numérico de Numerov aplicado à equação de Schroedinger. **Revista Brasileira de Ensino de Física**, v. 36, n. 2, 2310, 2014.

CARUSO, F. OGURI, SILVEIRA, F. **Revista Brasileira de Ensino de Física**, vol. 44, e20220098, 2022.

MENDONÇA, A. K. F. EVANGELISTA, T. S. RISPOLI, R. G. G. Amorim. Resolvendo numericamente equações diferenciais fracionárias. **Revista Brasileira de Ensino de Física**, vol. 44, e20210426, 2022.

QUINGA, S. Ecuaciones no lineales en física y su resolución mediante el uso de métodos iterativos multipaso de orden alto. **Revista de enseñanza de la física**, Vol. 33, no. 2, 2021.

CHICHARRO, F. **Análisis dinámico y aplicaciones de métodos iterativos de resolución de ecuaciones no lineales**. Valencia: Universitat Politécnica de Valencia, 2017.

JUSTO, D. A. R. et. al. **Cálculo Numérico Um Livro Colaborativo Versão Python** p.290, UFRGS, 2020.

HALLIDAY, D.; WALKER, J. RESNICK, R. **Fundamentos de Física,** 8. ed., Rio de Janeiro: LTC, 2009.

SHAMPINE, L. W. **"Some Practical Runge-Kutta Formulas", Mathematics of Computation**, Vol. 46, No. 173, pp. 135-150, 1986.

GRIFFITHS. D. **Eletrodinâmica**, 3ª Ed, Pearson Edition, São Paulo 2011.

OGATA, K. **Engenharia de Controle moderno**. 4.ed., Prentice-Hall, 2003.

APÊNDICES

#EDO do problema de Atwood

```
import matplotlib.pyplot as plt
def f(x,y):
   return (((m2-m1+k*x)*(g))/(m2+m1-k*x)+y*((k)/(m2+m1-k*x)))
# Condições de contorno
x0 = 0
y0 = 0
m1=2
g=9.80
m2=4
k=0.15

#Intervalo
a = 0
b = 20
N = 200
h = (b-a)/N #Valor do passo
#Vetores
xi = [x0]
yi = [y0]
```

```
#Iterações
for i in range(N):
    k1 = f(x0,y0)
    k2 = f(x0+h/2, y0+h/2*k1)
    k3 = f(x0+h/2, y0+h/2*k2)
    k4 = f(x0+h, y0 + h*k3)
    yk = y0 + (h/6)*(k1 + 2*(k2+k3) + k4)
    x0 = x0 + h
    y0 = yk
    print(f"({round(x0,6)} , {round(y0,6)})")
    #Anexando aos vetores
    xi.append(round(x0,6))
    yi.append(round(yk,6))

print()
print(xi)
print(yi)

import matplotlib.pyplot as plt
plt.plot(xi,yi,color="blue",linewidth=2.0)
plt.title("Grafico da velocidade em função do tempo", color="black")
plt.xlabel("Tempo (s)", color="black")
plt.ylabel("Velocidade (m/s)", color="black")
plt.plot(xi,yi)
plt.show()
```

#EDO do problema do campo magnético

```
import matplotlib.pyplot as plt
def f(x,y):
    return (g-((B**2)*(d**2)/(r*m))*y)
# Condições de contorno
x0 = 0
y0 = 0
B=0.75
g=9.81
m=0.05
d=0.30
r=4.7

#Intervalo
a = 0
b = 20
N = 200
h = (b-a)/N #Valor do passo
#Vetores
xi = [x0]
yi = [y0]
#Iterações
for i in range(N):
    k1 = f(x0,y0)
    k2 = f(x0+h/2, y0+h/2*k1)
    k3 = f(x0+h/2, y0+h/2*k2)
    k4 = f(x0+h, y0 + h*k3)
    yk = y0 + (h/6)*(k1 + 2*(k2+k3) + k4)
    x0 = x0 + h
    y0 = yk
    print(f"({round(x0,6)} , {round(y0,6)})")
    #Anexando aos vetores
    xi.append(round(x0,6))
```

```
    yi.append(round(yk,6))

print()
print(xi)
print(yi)

import matplotlib.pyplot as plt
plt.plot(xi,yi,color="blue",linewidth=2.0)
plt.title("Grafico da velocidade em função do tempo", color="black")
plt.xlabel("Tempo (s)", color="black")
plt.ylabel("Velocidade (m/s)", color="black")
plt.plot(xi,yi)
plt.show()
```