Instruções

Em grupos de 3 a 4 integrantes construa os algoritmos dos exercícios propostos.

Para o segundo exercício, vejam o notebook neste **link**, ou pela URL `https://colab.research.google.com/drive/1o0g7IUqmfW7HwhDGJWY4hD7GPOhno4Me?usp=sharing`. O notebook contém o tutorial do uso da função que é pedida no exercício 2.

Procurem construir as funções da forma mais genérica possível, para evitar repetição de código.

Sugestão:implemente os algoritmos do ex. 1 tendo em mente o uso do algoritmo apresentado no notebook, ou seja, construir um algoritmo que seja capaz de trabalhar com uma função de mesma estrutura que foi apresentada no notebook, i.e.,
Algoritmo pronto:

solve_ivp(**func**, t_interval, conds, t_eval=t).y.

Sugestão de construção:

NOME_SUA_FUNCAO(**func**, params...).

Exercício 1

Considere os seguintes sistemas de EDO:

a.
$$\begin{aligned} u_1' &= 3u_1 + 2u_2 - (2t^2+1)e^{2t}, & 0 \le t \le 1, \quad u_1(0) = 1; \\ u_2' &= 4u_1 + u_2 + (t^2+2t-4)e^{2t}, & 0 \le t \le 1, \quad u_2(0) = 1; \end{aligned}$$

b.
$$\begin{aligned} u_1' &= -4u_1 - 2u_2 + cos(t) + 4sen(t), & 0 \le t \le 2, \quad u_1(0) = 0 \\ u_2' &= 3u_1 + u_2 - 3sen(t), & 0 \le t \le 2, \quad u_2(0) = -1 \end{aligned}$$

c.
$$\begin{aligned} u_1' &= u_2, & 0 \le t \le 2, \quad u_1(0) = 1 \\ u_2' &= -u_1 - 2e^t + 1, & 0 \le t \le 2, \quad u_2(0) = 0 \\ u_3' &= -u_1 - e^t + 1, & 0 \le t \le 2, \quad u_1(0) = 1 \end{aligned}$$

d.
$$\begin{aligned} u_1' &= u_2 - u_3 + t, & 0 \le t \le 1, \quad u_1(0) = 1 \\ u_2' &= 3t^2, & 0 \le t \le 1, \quad u_2(0) = 1 \\ u_3' &= u_2 + e^{-t}, & 0 \le t \le 1, \quad u_1(0) = -1 \end{aligned}$$

Implemente e utilize os métodos de Euler explícito, melhorado(Runge-Kutta) e modificado(Runge-Kutta) para aproximar cada sistema de edo's apresentados anteriormente, use os valores de h especificados logo abaixo. Em seguida, compare o erro absoluto para cada aproximação de acordo com as soluções exatas dadas. Apresente em forma de tabelas os resultados comparando os erros de cada método numérico utilizado.

a. Use $h = 0.2$ (Dica: $h = 0.2 \Rightarrow$ np.linspace(0, 1, 6))

   Soluções:
   $u_1 = \frac{1}{3}e^{5t} - \frac{1}{3}e^{-t} + e^{2t}$
   $u_2 = \frac{1}{3}e^{5t} + \frac{2}{3}e^{-t} + t^2e^{2t}$

b. Use $h = 0.1$

   Soluções:
   $u_1 = 2e^{-t} - 2e^{-2t} + sen(t)$
   $u_2 = -3e^{-t} + 2e^{-2t}$

c. Use $h = 0.5$

   Soluções:
   $u_1 = cos(t) + sen(t) - e^t + 1$
   $u_2 = cos(t) - sen(t) - e^t$
   $u_3 = cos(t) - sen(t)$

d. Use $h = 0.1$

   Soluções:
   $u_1 = -0.05t^5 + 0.25t^4 + t + 2 - e^{-t}$
   $u_2 = t^3 + 1$
   $u_3 = 0.25t^4 + t - e^{-t}$

Exercício 2

A simulação da trajetória de uma bola de beisebol após o lançamento feito por um arremessador pode ser descrita pelas equações

$$\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t} = \mathbf{V}, \qquad \frac{\mathrm{d}\mathbf{V}}{\mathrm{d}t} = \mathbf{F},$$

Onde $\mathbf{x}(t) = (x(t), y(t), z(t))^T$ representa a posição atual da bola no instante t, $\mathbf{v} = (v_x, v_y, v_z)^T$ a velocidade e $\mathbf{F} = (F_x, F_y, F_z)^T$, sendo

$$\begin{aligned} F_x &= -F_{fric}(v)vv_x + B\omega(v_z sen(\phi) - v_y cos(\phi)), \\ F_y &= -F_{fric}(v)vv_y + B\omega v_x cos(\phi), \\ F_z &= -F_{fric}(v)vv_z - B\omega v_x sen(\phi) - g. \end{aligned}$$

$g =$ Gravidade (adote 9.8)

$v = \|\mathbf{v}\|$

$B = 4.1 \times 10^{-4}$

$\phi =$ Ângulo de lançamento

$\omega =$ Módulo da vel. angular aplicada no lançamento.

F_{fric} é o coeficiente de fricção definido, nesse caso, como

$$F_{fric}(v) = 0.0039 + \frac{0.0058}{1 + e^{0.2(v-35)}}. \tag{1}$$

Considere as seguintes condições iniciais: $x(0) = [0, 0, 0]^T$ e $v(0) = [38cos(1°), 0, 38sen(1°)]^T$. E também $\phi = 0°$ e $\omega = 180 \cdot 1.047198rad/s$.

Defina e construa o sistema de equações para ser utilizada na função ***solve_ivp*** do pacote *scipy* (utiliza Runge-Kutta com precisão de quarta ordem por padrão) para obter a solução desse problema e responda: Depois de quantos segundos, aproximadamente, a bola irá, tocar o solo, ou seja, $z = 0$? Considere $0 \leq t \leq 0.5$, e para o *linspace* utilize 51 divisões, isso resulta em um $h = 0.01$.
Use suas implementações e compare o erro entre elas e o resultado da função pronta, plote as aproximações e erros absolutos correspondentes.