Prova A1 – Equações Diferenciais Parciais

1) Descreva as equações das curvas características das EDPs abaixo

a) $u_t + u_x = 0$

b) $xu_t - tu_x = u$, com x e t no primeiro quadrante, isto é, $x \geq 0$ e $t \geq 0$

2) Encontre a solução explícita do item (1b) com dado inicial $u(x, 0) = x$

3) Considere o problema de valor inicial $u_t + u \cdot u_x = 0$,

com $u(x, 0) = \begin{cases} 10 & , se\ x \leq 0 \\ 1 & , se\ x > 0 \end{cases}$.

Descreva a função $h: R \to R$ dada por $h(x) = u(x, 5)$, onde $u(x, t)$ é a solução fraca do problema de valor inicial acima.

4) método numérico equação de transporte

Vamos implementar um método numérico (método de Euler no tempo) para encontrar uma aproximação do problema de valor inicial $\begin{cases} u_t + c \cdot u_x = 0 \\ u(x, 0) = f(x) \end{cases}$. Vamos supor que $c = 1$.

O código em Python em anexo implementa o método de Euler no tempo. A ideia é que a derivada espacial seja aproximada por $u_x \approx \frac{u(x+\Delta x) - u(x)}{\Delta x}$ ou, $u_x \approx \frac{u(x) - u(x-\Delta x)}{\Delta x}$. A primeira aproximação é chamada de discretização à direita da derivada e a segunda de discretização à esquerda. Ambas aproximam a derivada com a mesma ordem de precisão (mesmo erro local).

O código está quase pronto, faltando completar o passo de Euler, que deve incluir a derivada discretizada no espaço.

a) Descreva a solução exata $u(x, t)$ para $t = 0$, $t = 0.2$ e $t = 0.6$
b) Complete o código com a discretização à direita. Plote a solução numérica encontrada $U(x, t)$ para $t = 0$, $t = 0.2$ e $t = 0.6$
c) Complete o código com a discretização à esquerda. Plote a solução numérica encontrada $U(x, t)$ para $t = 0$, $t = 0.2$ e $t = 0.6$
d) Explique porque as soluções são tão diferentes, uma vez que as derivadas à direita e à esquerda são similares em termos de precisão da discretização. O que aconteceria com as soluções numéricas se $c = -1$?

```
import numpy as np
import matplotlib.pyplot as plt

numero_pontos_tempo = 50
numero_pontos_espaco = 50

c = 1 #velocidade da onda

t = np.linspace(0,1,numero_pontos_tempo)
x = np.linspace(0,1,numero_pontos_espaco)

dt = t[1]-t[0]
dx = x[1]-x[0]

f = np.sin(3.1415196*x) # valor inicial - você pode trocar essa funcao
u = np.zeros([numero_pontos_tempo, numero_pontos_espaco])

u[0,] = f #valor inicial
for tk in range(0,numero_pontos_espaco-1):
   for xk in range(numero_pontos_espaco):
      # ---  vizinhos do ponto xk: cuidado para garantir a periodicidade da funcao ---------
      xk_esquerda = xk-1
      if xk_esquerda < 0:
         xk_esquerda = numero_pontos_espaco-1
      xk_direita = xk+1
      if xk_direita >= numero_pontos_espaco:
         xk_direita = 0
      #--------

      #passo de euler ************ COMPLETAR AQUI **********************
      u[tk+1,xk] = u[tk, xk]+?
```