Lista 4

1) Determine o tipo das equações abaixo (parabólica, hiperbólica e elítica):
a) $u_t = u_{xx}$
b) $4u_{xx} + 12u_{xy} + 5u_{yy} = 6ux - uy$
c) $u_{xx} + 4u_{xy} + 4u_{yy} = 4 + 2u_x$
d) $(1 + x^2)u_{xx} + (1 + y^2)u_{yy} = 0$

2) Considere a equação da onda $u_{tt} = c^2 u_{xx}$, com dados iniciais $u(x, 0) = sen(2\pi x)$ e $u_t(x, 0) = 0$.
a) Encontre a solução deste PVI.
b) Mostre que, para cada t, a função $h(x) = u(x, t)$ é periódica e calcule o seu período.
b) Mostre que, para todo t, $u(0, t) = 0$ e $u(1, t) = 0$

3) Ainda com o problema de valor inicial anterior, $u_{tt} = c^2 u_{xx}$, com $u(x, 0) = sen(2\pi x)$ e $u_t(x, 0) = 0$, vamos escrever um programa em Python para encontrar uma aproximação numérica da solução $U(x, t)$ por diferenças finitas, no mesmo espírito do exercício da prova. A ideia é definir uma grade regular de pontos (x_k, t_k) e calcular aproximações numéricas da solução $U(x_k, t_k)$. Neste exercício, a grade de pontos está contida no retângulo $[0,1] \times [0,1]$, ou seja, $0 \leq x_0 \leq x_1 \leq \cdots \leq x_N < 1$ e $0 \leq t_0 \leq t_1 \leq \cdots \leq t_M < 1$.

Para a aproximação numérica, vamos usar o método de Euler, adaptado para equações de segunda ordem.

1 - Primeiro, a aproximação da derivada u_{xx} (já vimos essa aproximação antes...)

$$u_{xx}(x, t) \approx \frac{u(x - \Delta x, t) - 2u(x, t) + u(x + \Delta x, t)}{\Delta \mathrm{x}^2}$$

Agora os passos de Euler em duas etapas

2 - A velocidade u_t pode evoluir (aqui entra e a EDP)

$$u_t(x, t + \Delta t) \approx u_t(x, t) + \Delta t(-c^2 \cdot u_{xx}(x, t))$$

3 - Finalmente, a função u

$$u(x, t + \Delta t) \approx u(x, t) + \Delta t(u_t(x, t + \Delta t))$$

a) Complete o código em anexo para que ele reflita o método numérico descrito acima.
b) Plote a solução numérica encontrada para $t = 0$, $t = 0.2$ e $t = 0.9$
b) O número de pontos na discretização da variável t é 20 e na discretização de x também é 20. Vamos aumentar o número de pontos do grid na esperança de diminuir o erro de aproximação.

Passando o número de pontos da variável t para 50 e x para 100, o que ocorre com a simulação? O que acontece se, ao invés disso, trocarmos o número de pontos de t para 100 e de x para 50? Por que os resultados são tão diferentes?

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

numero_pontos_tempo = 20
numero_pontos_espaco = 20

c = 0.8 #velocidade da onda

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

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

f = np.sin(2*3.1415196*x) # valor inicial - você pode trocar essa funcao
g = np.zeros([numero_pontos_espaco]) # valor inicial

u = np.zeros([numero_pontos_espaco, numero_pontos_tempo])

u[:,0] = f #valor inicial
ut = g # valor inicial
for tk in range(0,numero_pontos_tempo-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
        #--------

        # COMPLETE AQUI
        # aproximacao de uxx
        #uxx = ???

        # COMPLETE AQUI
        # calcular ut
        # ut[xk] = ???

        # COMPLETE AQUI
        # calcular a funcao u
        #u[xk, tk+1] = ???

plt.plot(x,u[:, 0], 'blue')
plt.plot(x,u[:, 4], 'magenta')
plt.plot(x,u[:, 18], 'red')
```