Computação Científica II

Métodos numéricos para resolver equações diferenciais

2020-PLE

A-14

Exemplo 2

- Queremos resolver a equação de calor em estado estacionário

$$-\nabla(\kappa\nabla T) = 0$$

- Com dois tipos de condições de contorno

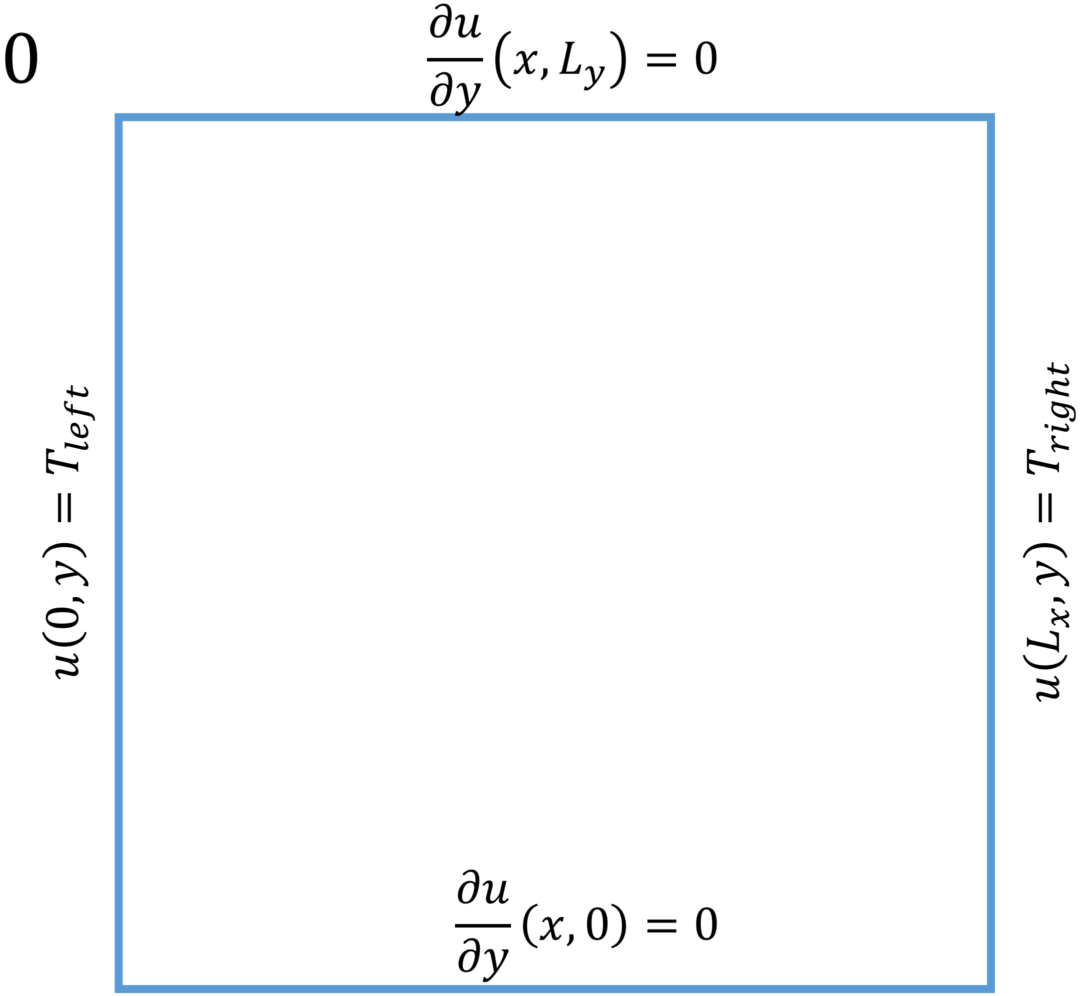
$$T_{left} = 500\ K$$

$$T_{right} = 293.0\ K$$

- Queremos considerar a condutividade

variável.

$$\kappa = k(x,y)$$



Exemplo 2

- Podemos discretizar a equação como

$$\frac{1}{h_x}\left(\kappa\left(x_{i+\frac{1}{2},j}\right)\frac{u_{i+1,j}-u_{i,j}}{h_x}-\kappa\left(x_{i-\frac{1}{2},j}\right)\frac{u_{i,j}-u_{i-1,j}}{h_x}\right)$$
$$+\frac{1}{h_y}\left(\kappa\left(x_{i,j+\frac{1}{2}}\right)\frac{u_{i,j+1}-u_{i,j}}{h_y}-\kappa\left(x_{i,j-\frac{1}{2}}\right)\frac{u_{i,j}-u_{i,j-1}}{h_y}\right)=0$$

- Condição de contorno de Neumann será

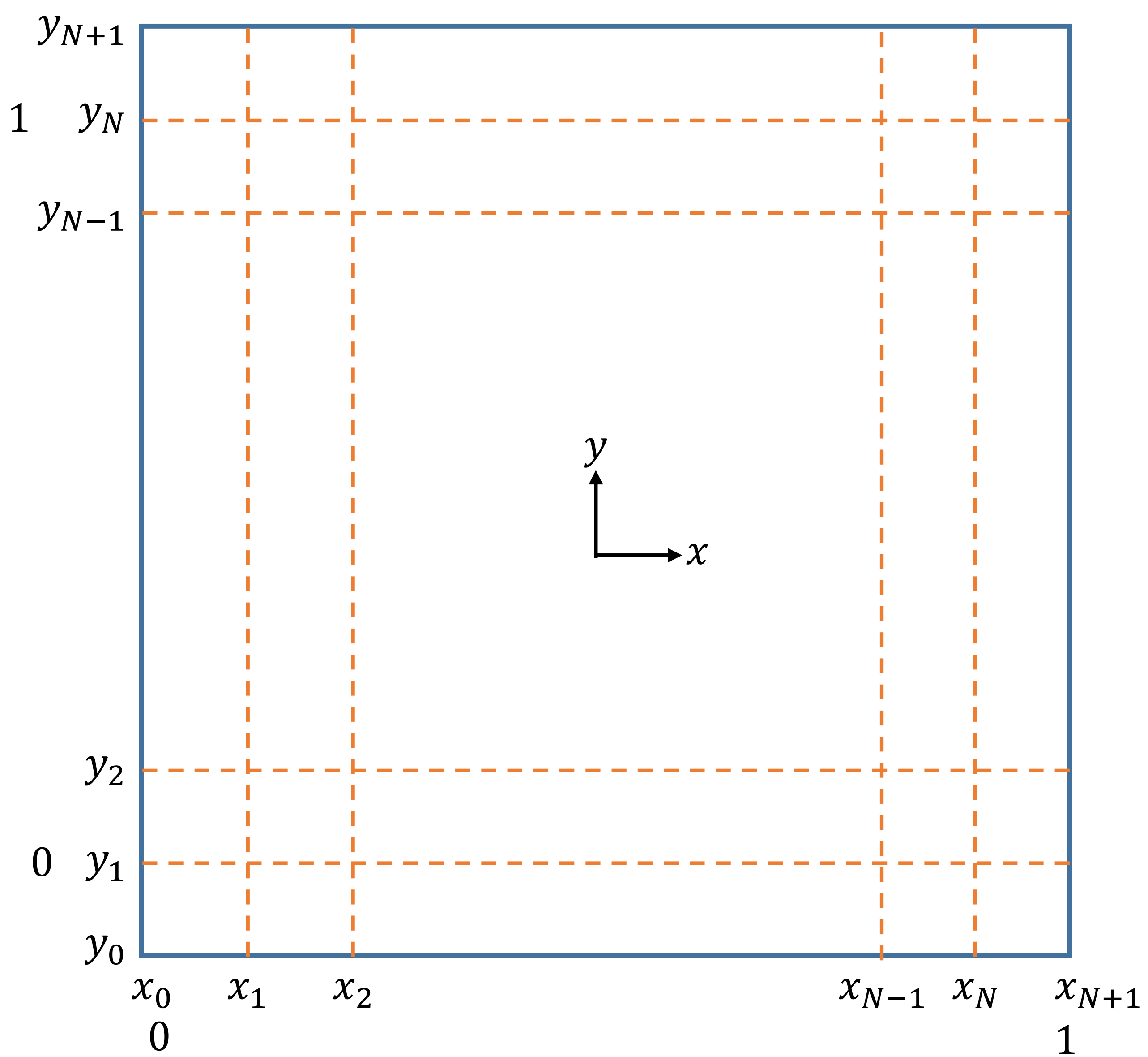
$$\frac{u_{i,j+1}-u_{i,j-1}}{2h_y}=0$$

Exemplo 2

- Primeiro precisamos definir a grade.
- Como no caso 1D, podemos definir o grade como

$$h_x = \frac{L_x}{N+1}$$
$$h_y = \frac{L_y}{N-1}$$

$$x_i = h_x \times i$$
$$y_j = h_y \times (j-1)$$



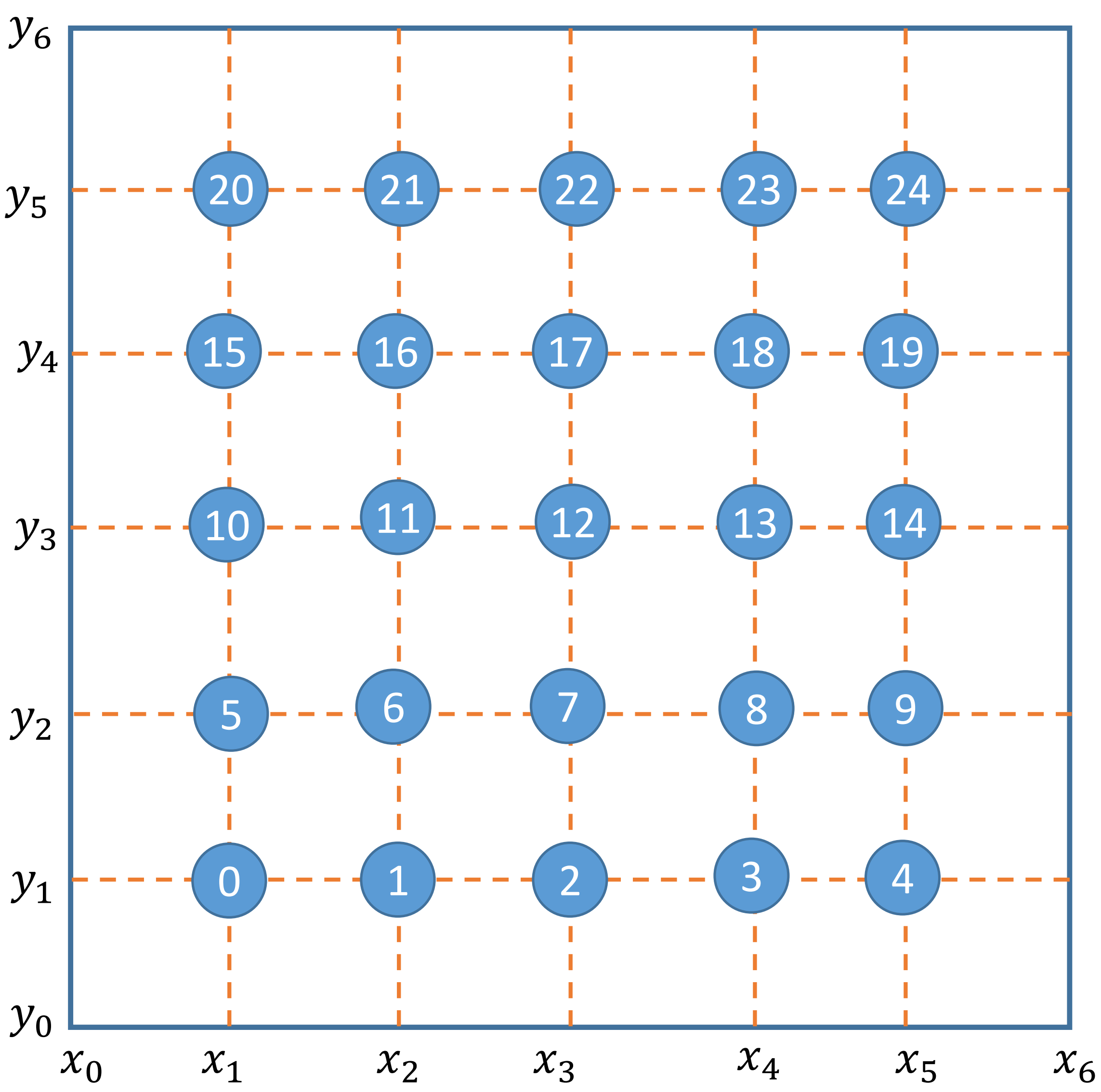
Renumeração

- É mais fácil usar n em vez (i, j) para identificar os nós.

$$n = (i - 1) + (j - 1) \times N_x$$

- Agora, a solução é

$$u_n = u_{i,j}$$



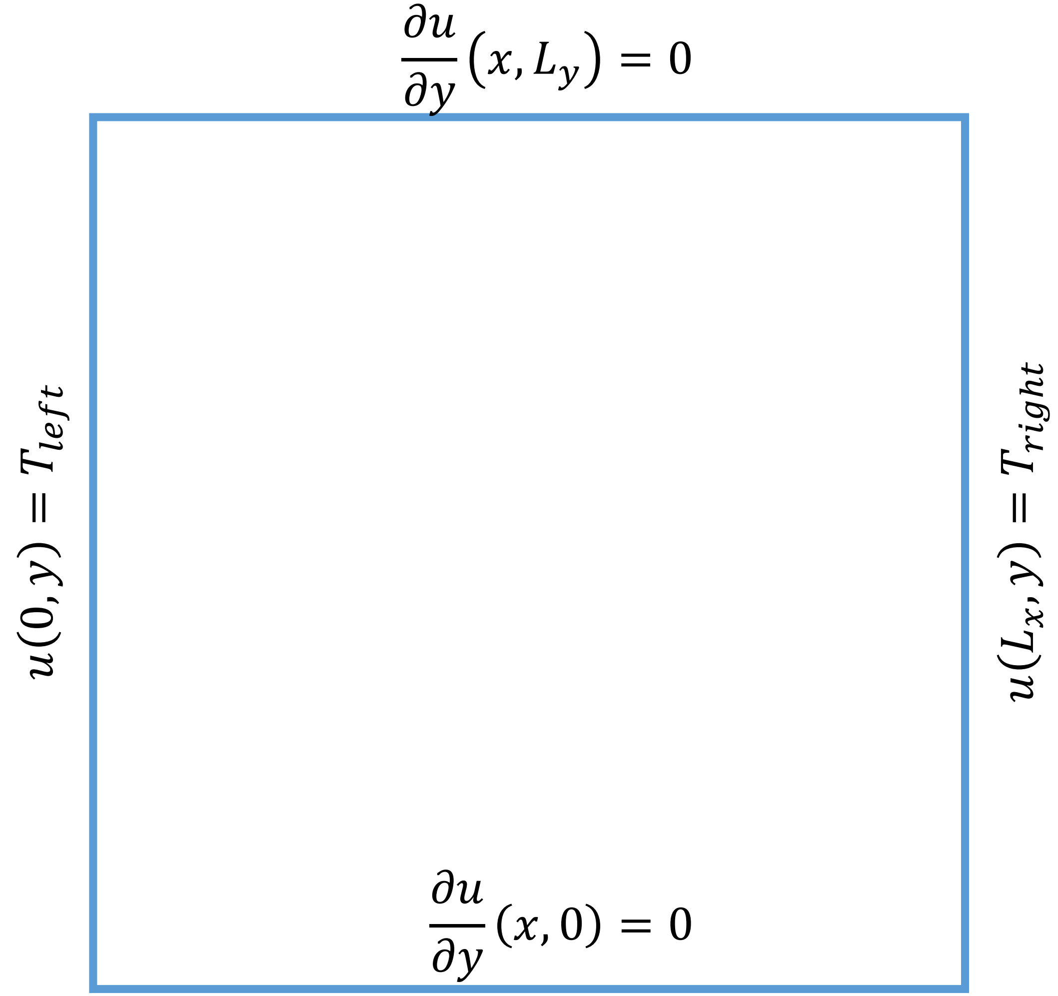
Exemplo 2

- Verifique a solução para três casos,

1. $k = 410.0$

2. $k = \begin{cases} 410 & x < 0.05 \\ 0.1 & x > 0.05 \end{cases}$

3. $k = \begin{cases} 410 & outside\ circle\ (0.05{,}0.03) \\ 0.1 & inside\ circle \end{cases}$



Equações Parabólicas

- Hoje vamos ver como podemos resolver equações parabólicas.
- Novamente, nós adiamos a definição exata da equação parabólica para o futuro.
- Só para saber, a equação de calor “unsteady” é um exemplo das equações parabólicas.

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2} \qquad 0 < x < 1, \quad 0 < t < t_e nd, \quad \kappa > 0$$

$$T(0,t) = \alpha(t), \quad T(1,t) = \beta(t), \quad T(x,0) = T_0(x)$$

Equações Parabólicas

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2}$$

- Nós temos tudo para resolver esse problema.
- Podemos usar a diferença central para o derivado espacial.

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{h_x^2}$$

- E podemos usar o método de Euler avançado para a derivada de tempo.

$$\frac{\partial T}{\partial t} = \frac{T_i^{n+1} - T_i^n}{h_t}$$

- Aqui T_i^n significa $T_i^n = T(x = x_i, t = t_n)$

Equações Parabólicas

- Portanto, temos

$$\frac{T_i^{n+1} - T_i^n}{h_t} = \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{h_x^2}$$

- Ou

$$T_i^{n+1} = \sigma T_{i-1}^n + (1 - 2\sigma) T_i^n + \sigma T_{i+1}^n \qquad \sigma = \frac{h_t}{h_x^2}$$

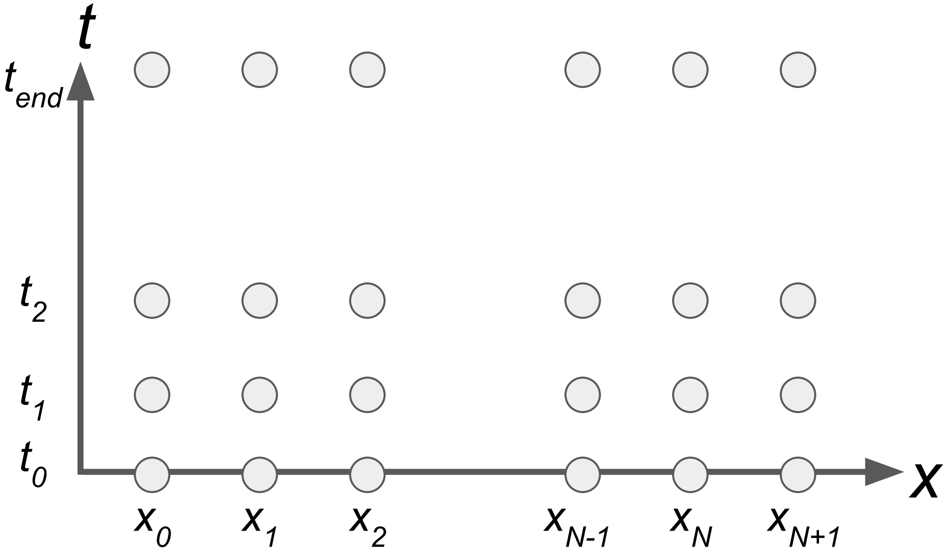
- Preste atenção, esta solução é explícita. Se tivermos condições iniciais e de contorno, podemos encontrar a temperatura a qualquer momento.

Equações Parabólicas

$$T_i^{n+1} = \sigma T_{i-1}^n + (1 - 2\sigma)T_i^n + \sigma T_{i+1}^n \qquad \sigma = \frac{h_t}{h_x^2}$$

- Além disso, semelhante à equação elíptica 2d que estudamos antes, aqui temos uma grade 2d.
- Nós chamamos esse método FTCS.

Forward-Time-Central-Space

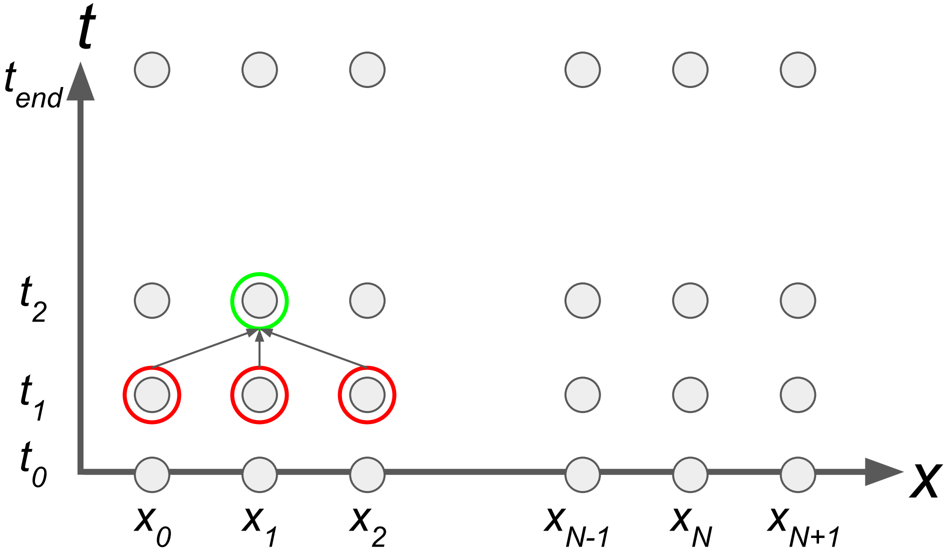


Equações Parabólicas

$$T_i^{n+1} = \sigma T_{i-1}^n + (1-2\sigma)T_i^n + \sigma T_{i+1}^n \qquad \sigma = \frac{h_t}{h_x^2}$$

- Para calcular cada ponto no novo intervalo de tempo, precisamos de informações de três pontos no passo de tempo anterior.

Stencil



Exercício 1

- Queremos resolver o seguinte problema

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2} \qquad 0 < x < 1, \quad 0 < t < t_e nd, \quad \kappa > 0$$

$$T(0,t) = \alpha(t), \quad T(1,t) = \beta(t), \quad T(x,0) = T_0(x)$$

- Primeiro, resolva essas configurações

$$\alpha(t) = 0, \quad \beta(t) = 0,$$
$$T_0(x) = sin(\pi x), \quad \kappa = 1$$

Jupyter-Notebook (Animation)

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import animation, rc
from IPython.display import HTML

# First set up the figure, the axis, and the plot element we want to animate
fig, ax = plt.subplots()
ax.set_xlim(( 0, 2))
ax.set_ylim((-2, 2))
line, = ax.plot([], [], lw=2)

# initialization function: plot the background of each frame
def init():
    line.set_data([], [])
    return (line,)

# animation function. This is called sequentially
def animate(i):
    x = np.linspace(0, 2, 1000)
    y = np.sin(2 * np.pi * (x - 0.01 * i))
    line.set_data(x, y)
    return (line,)

# call the animator. blit=True means only re-draw the parts that have changed.
anim = animation.FuncAnimation(fig, animate,
init_func=init,frames=100, interval=20, blit=True)

HTML(anim.to_html5_video())
```

Exercício 1

- Queremos resolver o seguinte problema

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2} \qquad 0 < x < 1, \quad 0 < t < t_end, \quad \kappa > 0$$

$$T(0,t) = \alpha(t), \quad T(1,t) = \beta(t), \quad T(x,0) = T_0(x)$$

- Agora, resolva essas configurações

$$\alpha(t) = 0, \quad \beta(t) = 1,$$
$$T_0(x) = \left|sin(\tfrac{3\pi x}{2})\right|, \quad \kappa = 1$$

Método das Linhas

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2}$$

- Vamos usar a diferença central para discretizar a derivada espacial, mas manter a derivada temporal inalterada.

$$\frac{\partial T}{\partial t}(t) = \frac{T_{i+1}(t) - 2T_i(t) + T_{i-1}(t)}{h_x^2} \quad T_i(0) = T_0(x_i) \quad i = 1, 2, \ldots, N$$

- Nós chamamos essa equação semi-discretizada.
- Os limites serão

$$\frac{\partial T}{\partial t}(t) = \frac{T_2(t) - 2T_1(t) + \alpha(t)}{h_x^2}$$

$$\frac{\partial T}{\partial t}(t) = \frac{\beta(t) - 2T_N(t) + T_{N-1}(t)}{h_x^2}$$

Método das Linhas

$$\frac{\partial T}{\partial t}(t) = \frac{T_{i+1}(t) - 2T_i(t) + T_{i-1}(t)}{h_x^2} \qquad T_i(0) = T_0(x_i) \quad i = 1, 2, \ldots, N$$

$$\frac{\partial T}{\partial t}(t) = \frac{T_2(t) - 2T_1(t) + \alpha(t)}{h_x^2} \qquad \frac{\partial T}{\partial t}(t) = \frac{\beta(t) - 2T_N(t) + T_{N-1}(t)}{h_x^2}$$

- Podemos escrever essas equações como um ODE de primeira ordem.

$$\frac{d\mathbf{u}}{dt} = A\mathbf{u} + \mathbf{b}(t), \qquad \mathbf{u}(0) = \mathbf{u}_0$$

$$\mathbf{u} = [T_1, T_2, \ldots, T_N]^T$$

$$\mathbf{u}_0 = [T_0(x_1), \ldots, T_0(x_N)]^T$$

$$\mathbf{b}(t) = \frac{1}{h_x^2}[\alpha(t), 0, \ldots, 0, \beta(t)]^T$$

$$A = \frac{1}{h_x^2}\begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 1 & -2 & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{bmatrix}$$

Método das Linhas

- Vamos focar na matriz A. Nós podemos escrever

$$A = \frac{1}{h_x^2} tridiag(1, -2, 1)$$

- No capítulo 4, conhecemos os autovalores dessa matriz.

$$\lambda_j(A) = -\frac{4}{h_x^2} sin^2\left(\frac{j\pi}{2(N+1)}\right), \quad j = 1, \ldots, N$$

- Da "grid" nós sabemos $h_x(N+1) = 1$. Portanto, temos

$$\lambda_j(A) = -\frac{4}{h_x^2} sin^2\left(\frac{j\pi h_x}{2}\right), \quad j = 1, \ldots, N$$

Método das Linhas

- Todos os autovalores são reais, negativos e de magnitude muito diferente.

$$\lambda_j(A) = -\frac{4}{h_x^2} sin^2\left(\frac{j\pi h_x}{2}\right), \quad j = 1, .., N$$

$$A = \frac{1}{h_x^2}\begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 1 & -2 & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{bmatrix}$$

$$\lambda_1(A) \approx -\pi^2$$

$$\lambda_N(A) \approx -\frac{4}{h_x^2}$$

Equação rígida!

Método das Linhas

$$\frac{d\mathbf{u}}{dt} = A\mathbf{u} + \mathbf{b}(t), \quad \mathbf{u}(0) = \mathbf{u}_0$$

- Vamos resolver isso usando o método de Euler avançado.

$$\mathbf{u}^{n+1} = \mathbf{u}^n + h_t\left(A\mathbf{u}^n + \mathbf{b}(t_n)\right)$$

- No capítulo 3, vimos que para este sistema o método euler é estável se

$$h_t\lambda_j \in S_{FE} \quad \rightarrow \quad h_t\frac{4}{h_x^2} < 2 \quad \rightarrow \quad c = \frac{h_t}{h_x^2} < \frac{1}{2}$$

- Agora, repita o exercício 1 para ***c* = *1*** e ***1/4***.

Exercícios

- Exercício 6.3.3