Solução Numérica de Equações Diferenciais Ordinárias

EMC410235 - Programação Científica para Engenharia e Ciência Térmicas

Prof. Rafael F. L. de Cerqueira

2025.2

Objetivos da Aula

- Revisar conceitos fundamentais de EDOs e sua aplicação em engenharia térmica
- Resolver numericamente Problemas de Valor Inicial (PVI) com:
  - Métodos de Euler, Ponto Médio e Runge-Kutta
- Explorar a biblioteca `scipy.integrate` e o uso de `solve_ivp`
- Discutir problemas rígidos (stiff) e métodos implícitos
- Resolver sistemas de EDOs acopladas
- Introduzir Problemas de Valor de Contorno (PVC)
- Aplicar o Método do Tiro a PVCs lineares e não lineares
- Preparar o terreno para diferenças finitas e EDPs na próxima aula

Equações Diferenciais Ordinárias (EDOs)

Definição

Uma equação diferencial ordinária (EDO) é uma equação que envolve derivadas de uma função incógnita em relação a **uma única variável independente**.

Exemplo de 1ª ordem:

$$\frac{dy}{dt} = -ky$$

Exemplo de 2ª ordem:

$$m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = 0$$

EDO vs EDP

- **EDO:** envolve derivadas em relação a uma única variável independente (geralmente tempo)
- **EDP:** envolve derivadas parciais em relação a duas ou mais variáveis (ex: espaço e tempo)

Exemplo de EDO:

$$\frac{dT}{dt} = -k(T - T_\infty)$$

Exemplo de EDP:

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

EDOs em Ciências Térmicas e Engenharia

- **Resfriamento de corpos (Lei de Newton):**

$$\frac{dT}{dt} = -hA(T - T_\infty)/\rho c_p V$$

- **Problema de condução unidimensional em regime permanente**

$$\frac{d^2T}{dx^2} = 0$$

- **Oscilador massa-mola-amortecedor:**

$$m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = F(t)$$

Método de Euler

Ideia central

Dado um problema de valor inicial:

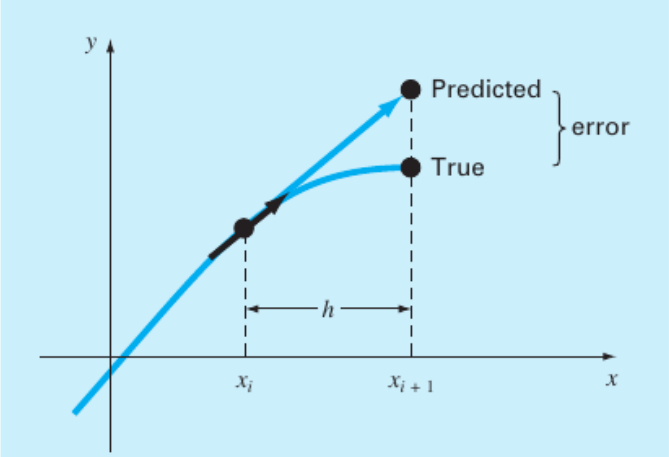
$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0$$

Aproximamos a solução com:

$$y_{n+1} = y_n + h \cdot f(t_n, y_n)$$

- h é o passo de tempo
- $f(t_n, y_n)$ é a inclinação no ponto (t_n, y_n)
- A solução é construída ponto a ponto

Interpretação Gráfica do Método de Euler



- A derivada indica a inclinação da reta tangente
- O método de Euler "caminha" pela tangente
- A precisão depende de h e da curvatura da solução

Exemplo: Método de Euler

- Resolver numericamente a EDO:

$$\frac{dy}{dx} = -2x^3 + 12x^2 - 20x + 8.5$$

- Intervalo: $x \in [0, 4]$
- Passo: $h = 0.5$
- Condição inicial: $y(0) = 1$
- Método: Euler

Solução exata:

$$y(x) = -0.5x^4 + 4x^3 - 10x^2 + 8.5x + 1$$

Fontes de Erro em Métodos Numéricos

Dois tipos principais de erro ao resolver EDOs numericamente:

1. **Erro de arredondamento**
   - Decorre da representação finita dos números em ponto flutuante.
   - É acumulativo: cada passo da integração carrega pequenos erros.
   - Costuma ser pequeno, mas pode se acumular em muitos passos ou em problemas mal condicionados.

2. **Erro de truncamento**
   - Decorre da aproximação da solução exata por uma fórmula numérica.
   - No método de Euler, vem da truncagem da série de Taylor após o primeiro termo.
   - É o erro dominante na maioria das aplicações práticas.

Origem dos Erros no Método de Euler

Fonte principal de erro: Erro de truncamento local

- O método de Euler usa uma **expansão de Taylor de primeira ordem**:

$$y(x+h) \approx y(x) + h \cdot y'(x)$$

- Ao truncar a série em apenas um termo, ignora-se:

$$\frac{h^2}{2} y''(\xi), \quad \text{com } \xi \in [x, x+h]$$

- Isso introduz um erro de truncamento local de ordem:

$$\mathcal{O}(h^2)$$

- O erro de truncamento global acumulado ao longo da integração é:

$$\mathcal{O}(h)$$

Consequência: Reduzir h melhora a precisão, mas aumenta o custo computacional.

Métodos de Taylor de Ordem Superior

A fórmula de Taylor de segunda ordem para resolver uma EDO é:

$$y_{i+1} = y_i + f(x_i, y_i)h + \frac{f'(x_i, y_i)}{2!}h^2$$

Com erro local de truncamento:

$$E_a = \frac{f''(x_i, y_i)}{6}h^3$$

Observação: para EDOs não triviais, é necessário aplicar a regra da cadeia ao derivar $f(x, y)$.

Derivadas no Método de Taylor – Regra da Cadeia

Primeira derivada de $f(x, y)$:

$$f'(x_i, y_i) = \frac{\partial f(x, y)}{\partial x} + \frac{\partial f(x, y)}{\partial y} \cdot \frac{dy}{dx}$$

Segunda derivada de $f(x, y)$:

$$f''(x_i, y_i) = \frac{\partial}{\partial x}\left[\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dx}\right] + \frac{\partial}{\partial y}\left[\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dx}\right] \cdot \frac{dy}{dx}$$

Conclusão: embora seja viável para funções simples (e.g. polinômios), o uso de métodos de Taylor de ordem superior se torna impraticável para funções mais gerais.

Método do Ponto Médio

Objetivo: melhorar a precisão do método de Euler ao estimar a inclinação no **meio do intervalo**.

Passos:

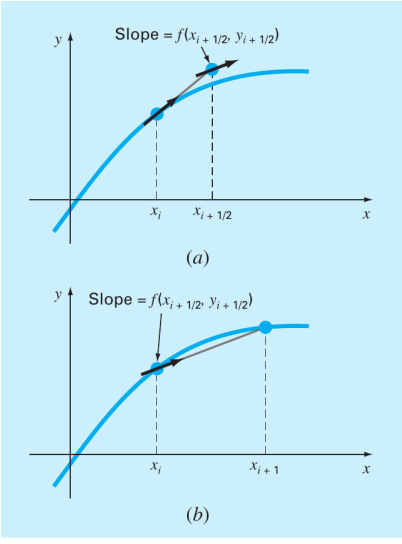
1. Calcular a inclinação provisória:
$$k_1 = f(x_n, y_n)$$
2. Estimar o ponto médio:
$$x_{n+1/2} = x_n + \frac{h}{2}, \quad y_{n+1/2} = y_n + \frac{h}{2}k_1$$
3. Calcular a inclinação no ponto médio:
$$k_2 = f(x_{n+1/2}, y_{n+1/2})$$
4. Atualizar a solução:
$$y_{n+1} = y_n + h \cdot k_2$$

Interpretação Gráfica do Método do Ponto Médio

- Em vez de seguir a tangente no início do intervalo (como em Euler),
- O método usa a derivada no ponto médio para avançar com mais precisão.
- Isso reduz o erro de truncamento:

Erro local: $\mathcal{O}(h^3)$

Erro global: $\mathcal{O}(h^2)$



Métodos de Runge-Kutta – Motivação

- Os métodos de Taylor de ordem superior são precisos, mas exigem derivadas sucessivas complicadas.
- Métodos de Runge-Kutta fornecem maior precisão **sem precisar calcular derivadas de ordem superior**.
- A ideia central é **avaliar a função em diferentes pontos dentro do intervalo** para obter uma média ponderada das inclinações.

Exemplos:

- Método do ponto médio: Runge-Kutta de 2ª ordem
- RK4: Runge-Kutta de 4ª ordem – mais usado na prática

Método de Runge-Kutta de 4ª Ordem (RK4)

Dado o PVI:

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0$$

Os incrementos são calculados como:

$$\begin{aligned}
k_1 &= f(t_n, y_n) \\
k_2 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \\
k_3 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right) \\
k_4 &= f\left(t_n + h, y_n + hk_3\right)
\end{aligned}$$

Atualização da solução:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Implementação do RK4 em Python

```
def dydx(x, y):
        return -2*x**3 + 12*x**2 - 20*x + 8.5

h = 0.5
x = np.arange(0, 4.5, h)
y_rk4 = np.zeros_like(x)
y_rk4[0] = 1

for i in range(len(x)-1):
        k1 = dydx(x[i], y_rk4[i])
        k2 = dydx(x[i] + h/2, y_rk4[i] + h/2 * k1)
        k3 = dydx(x[i] + h/2, y_rk4[i] + h/2 * k2)
        k4 = dydx(x[i] + h, y_rk4[i] + h * k3)
        y_rk4[i+1] = y_rk4[i] + h/6 * (k1 + 2*k2 + 2*k3 + k4)
```

Usando solve_ivp (SciPy)

solve_ivp é a função recomendada do `scipy.integrate` para resolver problemas de valor inicial (PVI).

```
solve_ivp(fun, t_span, y0, method='RK45', t_eval=None)
```

Parâmetros principais:

- `fun(t, y)` – função que define a EDO
- `t_span` – tupla com tempo inicial e final (ex: `(0, 4)`)
- `y0` – condição inicial (ex: `[1.0]`)
- `method` – método numérico: `'RK45'`, `'RK23'`, `'Radau'`, `'BDF'`, `'LSODA'`
- `t_eval` – pontos em que deseja avaliar a solução

Saída: objeto com:

- `sol.t` – valores de tempo
- `sol.y` – solução numérica (vetor/matriz)
- `sol.nfev` – número de chamadas da função

Sistemas de Equações Diferenciais Ordinárias

Motivação:

- Muitos problemas da engenharia e da física envolvem múltiplas variáveis acopladas — ex: posição e velocidade, temperatura e fluxo de calor, concentrações químicas, etc.
- Também surgem naturalmente ao **transformar EDOs de ordem superior** em sistemas de primeira ordem.

Forma geral de um sistema:

$$\begin{cases} \frac{dy_1}{dt} = f_1(t, y_1, y_2, \ldots, y_n) \\ \frac{dy_2}{dt} = f_2(t, y_1, y_2, \ldots, y_n) \\ \vdots \\ \frac{dy_n}{dt} = f_n(t, y_1, y_2, \ldots, y_n) \end{cases}$$

Forma vetorial:

$$\frac{d\vec{y}}{dt} = \vec{f}(t, \vec{y}) \quad \text{com} \quad \vec{y}(t_0) = \vec{y}_0$$

Método de Euler para um Sistema de EDOs

Sistema:

$$\begin{cases} \frac{dy_1}{dx} = y_2 \\ \frac{dy_2}{dx} = -2y_2 - 5y_1 \end{cases} \qquad \text{com } y_1(0) = 1, \quad y_2(0) = 0$$

Implementação com Euler:

```
h = 0.1
x = np.arange(0, 10+h, h)
y1 = np.zeros_like(x)
y2 = np.zeros_like(x)
y1[0] = 1
y2[0] = 0

for i in range(len(x)-1):
        y1[i+1] = y1[i] + h * y2[i]
        y2[i+1] = y2[i] + h * (-2*y2[i] - 5*y1[i])
```

Resolvendo com solve_ivp

```
from scipy.integrate import solve_ivp

def sistema(x, Y):
        y1, y2 = Y
        return [y2, -2*y2 - 5*y1]

x_eval = np.arange(0, 10, h)
sol = solve_ivp(sistema, [0, 10], [1, 0], t_eval=x_eval)
```

O que é Rigidez (Stiffness)?

Rigidez ocorre em sistemas de EDOs que envolvem múltiplas escalas de tempo — uma parte da solução varia rapidamente, enquanto outra muda lentamente.

Consequência: métodos explícitos (como Euler ou RK4) precisam de passos extremamente pequenos para capturar a parte rápida, mesmo que estejamos interessados apenas no comportamento lento.

Analogia física:

- Imagine uma mola muito rígida (com constante elástica muito alta) acoplada a uma estrutura mais lenta.
- Para simular o sistema com estabilidade, os métodos explícitos exigem **passos pequenos demais** — a simulação se torna ineficiente.

Soluções:

- Usar métodos implícitos, que são mais estáveis para problemas stiff.
- Ex: `solve_ivp(method='Radau')` ou `'BDF'`

Exemplo Clássico de Rigidez

EDO:

$$\frac{dy}{dt} = -1000y + 3000 - 2000e^{-t}, \quad y(0) = 0$$

Solução exata:

$$y(t) = 3 - 0.998e^{-1000t} - 2.002e^{-t}$$

Comportamento:

- e^{-1000t}: decai muito rápido (componente rápida)
- e^{-t}: decai lentamente (componente lenta)
- A solução converge suavemente para 3, mas métodos explícitos **explodem** se o passo for muito grande.

Mesmo que a solução pareça suave, a presença de componentes rápidas exige métodos robustos.

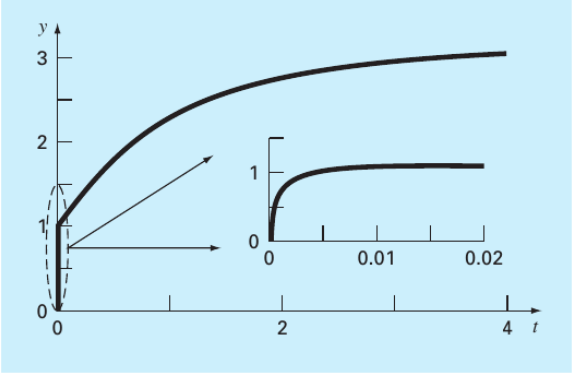
Exemplo Clássico de Rigidez

EDO:

$$\frac{dy}{dt} = -1000y + 3000 - 2000e^{-t}, \quad y(0) = 0$$

Solução exata:

$$y(t) = 3 - 0.998e^{-1000t} - 2.002e^{-t}$$



Euler aplicado a um problema rígido

EDO:

$$\frac{dy}{dt} = -1000y + 3000 - 2000e^{-t}, \quad y(0) = 0$$

Implementação com Euler:

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

def f(t, y):
        return -1000*y + 3000 - 2000*np.exp(-t)

h = 0.01
t = np.arange(0, 0.1 + h, h)
y = np.zeros_like(t)
y[0] = 0

for i in range(len(t) - 1):
        y[i+1] = y[i] + h * f(t[i], y[i])
```

Por que métodos explícitos falham em problemas rígidos?

Causa: instabilidade numérica.

Considere o modelo linear:

$$\frac{dy}{dt} = \lambda y, \quad \lambda < 0$$

O método de Euler para esse caso:

$$y_{n+1} = y_n + h\lambda y_n = (1 + h\lambda) y_n$$

Critério de estabilidade:

$$|1 + h\lambda| < 1 \quad \Rightarrow \quad h < \frac{2}{|\lambda|}$$

Problema rígido: $\lambda = -1000 \Rightarrow h < 0.002$

Conclusão:

- Métodos explícitos exigem passos absurdamente pequenos.
- Mesmo uma solução suave ($y(t) \to 3$) explode numericamente.

Implementação Implícita do Método de Euler

Ideia: A forma implícita (ou "para trás") de Euler avalia a derivada no ponto futuro:

$$y_{i+1} = y_i + h \cdot f(t_{i+1}, y_{i+1})$$

Exemplo: Para a EDO rígida

$$\frac{dy}{dt} = -1000y + 3000 - 2000e^{-t}$$

o método implícito resulta em:

$$y_{i+1} = y_i + h(-1000y_{i+1} + 3000 - 2000e^{-t_{i+1}})$$

Como é linear em y_{i+1}, podemos isolar:

$$y_{i+1} = \frac{y_i + 3000h - 2000he^{-t_{i+1}}}{1 + 1000h}$$

Vantagem: Método estável mesmo com passos grandes (estabilidade incondicional).

Classificação dos Métodos no **solve_ivp**

Método	Tipo	Ordem	Adaptativo	Comentários
`RK45`	Explícito	5(4)	Sim	Dormand-Prince. Bom para problemas não-stiff. Padrão do `solve_ivp`.
`RK23`	Explícito	3(2)	Sim	Runge-Kutta de ordem menor. Útil quando alta precisão não é necessária.
`DOP853`	Explícito	8	Sim	RK de alta ordem. Bom para problemas suaves. Mais caro.
`Radau`	Implícito	5	Sim	RK implícito. Muito estável. Indicado para problemas stiff.
`BDF`	Implícito	1–5	Sim	Fórmulas de Diferenciação Recuada (multistep). Bom para stiff.

Problemas de Valor de Contorno (PVC)

Problema de valor inicial (PVI):

$$y'(x) = f(x, y), \quad y(x_0) = y_0$$

Problema de valor de contorno (PVC):

$$y''(x) = f(x, y, y'), \quad \text{com } y(a) = \alpha, \quad y(b) = \beta$$

Exemplos clássicos:

- Barra com condução de calor estacionária
- Flexão de vigas com condições em ambas as extremidades
- Escoamento de Poiseuille entre placas planas

Abordagens numéricas:

- Método do tiro (reduz a um PVI)
- Método de diferenças finitas (discretiza o domínio)

Exemplo de PVC: Barra com Troca de Calor com o Ambiente

Problema: determinar o perfil de temperatura $T(x)$ em uma barra submetida a:

- Condução ao longo do comprimento
- Troca de calor por convecção com o ambiente ao redor

Equação diferencial:

$$\frac{d^2T}{dx^2} + h'(T_a - T) = 0 \quad \text{com} \quad h' = \frac{hP}{kA}$$

Condições de contorno típicas:

$$T(0) = T_0, \quad T(L) = T_L$$

Interpretação:

- É um problema de valor de contorno (PVC), pois os valores são especificados em ambas as extremidades.
- Aparece em aletas de dissipação térmica, paredes com perdas por convecção, trocadores de calor.

Exemplo de PVC: Barra com Troca de Calor com o Ambiente

Problema: determinar o perfil de temperatura $T(x)$ em uma barra submetida a:

- Condução ao longo do comprimento
- Troca de calor por convecção com o ambiente ao redor

Equação diferencial:

$$\frac{d^2T}{dx^2} + h'(T_a - T) = 0 \quad \text{com} \quad h' = \frac{hP}{kA}$$

Condições de contorno típicas:

$$T(0) = T_0, \quad T(L) = T_L$$

Interpretação:

- É um problema de valor de contorno (PVC), pois os valores são especificados em ambas as extremidades.
- Aparece em aletas de dissipação térmica, paredes com perdas por convecção, trocadores de calor.

Exemplo Numérico – Barra com Troca de Calor com o Ar

Problema físico: uma barra de $L = 10\,\text{m}$ dissipa calor para o ar ambiente.

Equação:

$$\frac{d^2T}{dx^2} + h'(T_a - T) = 0 \quad \text{com} \quad h' = 0{,}01$$

Condições de contorno:

$$T(0) = 40^\circ\text{C}, \quad T(10) = 200^\circ\text{C}, \quad T_a = 20^\circ\text{C}$$

Solução analítica:

$$T(x) = 73{,}4523\, e^{0{,}1x} - 53{,}4523\, e^{-0{,}1x} + 20$$

Objetivo: Resolver numericamente esse PVC usando o método do tiro, comparando com a solução analítica.

Método do Tiro (Shooting Method)

Ideia: transformar um problema de valor de contorno (PVC) em um problema de valor inicial (PVI).

Etapas:

1. Reescreva a EDO de 2ª ordem como um sistema de 1ª ordem.
2. Substitua a condição de contorno desconhecida (ex: $T'(0)$) por um chute: s.
3. Resolva o sistema com esse chute como se fosse um PVI.
4. Compare o valor calculado com a condição desejada na outra extremidade.
5. Ajuste s até satisfazer $T(L) = T_2$.

Método do Tiro (Shooting Method)

Ideia: transformar um problema de valor de contorno (PVC) em um problema de valor inicial (PVI).

Etapas:

1. Reescreva a EDO de 2ª ordem como um sistema de 1ª ordem.
2. Substitua a condição de contorno desconhecida (ex: $T'(0)$) por um chute: s.
3. Resolva o sistema com esse chute como se fosse um PVI.
4. Compare o valor calculado com a condição desejada na outra extremidade.
5. Ajuste s até satisfazer $T(L) = T_2$.

Método numérico:

- Utiliza métodos de PVI como `solve_ivp`.
- Ajuste de s pode ser feito com `scipy.optimize.root` ou `fsolve`.

Redução da EDO para sistema de 1ª ordem

Equação original:

$$\frac{d^2T}{dx^2} + h'(T_a - T) = 0$$

Forma reorganizada:

$$\frac{d^2T}{dx^2} = h'(T - T_a)$$

Variáveis auxiliares:

$$T_1(x) = T(x)$$

$$T_2(x) = \frac{dT}{dx}$$

Sistema equivalente:

$$\frac{dT_1}{dx} = T_2$$

$$\frac{dT_2}{dx} = h'(T_1 - T_a)$$

Condições:

- $T_1(0) = 40°C$
- $T_2(0) = s$ (chute)
- Buscar s tal que $T_1(10) = 200°C$

Método do Tiro com solve_ivp – chute manual

```python
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

# Parâmetros do problema
h_ = 0.01
Ta = 20
L = 10

# Sistema de 1ª ordem
def sistema(x, Y):
        T1, T2 = Y
        dT1dx = T2
        dT2dx = h_ * (T1 - Ta)
        return [dT1dx, dT2dx]

# Condições iniciais: T(0) = 40, T'(0) = s (chute)
s = 5.0  # valor inicial arbitrário para o chute
```

Método do Tiro para Problemas Não Lineares

Problema: barra aquecida com perdas por radiação (não linearidade simplificada)

Equação diferencial:

$$\frac{d^2T}{dx^2} + h_0(T - T_a)^4 = 0, \quad h_0 = 5 \times 10^{-8}$$

Condições de contorno:

$$T(0) = 40^\circ C, \quad T(10) = 200^\circ C, \quad T_a = 20^\circ C$$

Redução para sistema de 1ª ordem:

$$\begin{cases} \frac{dT}{dx} = z \\ \frac{dz}{dx} = h_0(T - T_a)^4 \end{cases}$$

Solução:

- Sistema pode ser resolvido com `solve_ivp`
- Aplicamos o método do tiro, ajustando o chute $z(0)$ para satisfazer $T(10) = 200^\circ C$
- O método continua válido mesmo com a não linearidade no termo fonte

Resolvendo PVCs com `solve_bvp` (SciPy)

`solve_bvp` é uma função da biblioteca `scipy.integrate` para resolver problemas de valor de contorno diretamente, sem necessidade do método do tiro.

Uso típico:

- Reescreva a EDO de ordem n como um sistema de 1ª ordem.
- Forneça uma função com as condições de contorno em ambas as extremidades.
- Especifique um chute inicial para a malha e para a solução.

Vantagens:

- Resolve PVCs **lineares ou não lineares**.
- Evita o processo iterativo do método do tiro.
- Ideal para problemas bem condicionados e com malhas conhecidas.

Documentação: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_bvp.html`

Método das Diferenças Finitas (MDF)

Ideia: discretizar a EDO no espaço e aproximar as derivadas por diferenças finitas.

Para a EDO da barra:

$$\frac{d^2T}{dx^2} + h'(T_a - T) = 0 \quad \Rightarrow \quad \frac{d^2T}{dx^2} = h'(T - T_a)$$

Aproximação da derivada:

$$\frac{d^2T}{dx^2} \approx \frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2}$$

Substituindo na equação:

$$\frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2} = h'(T_i - T_a)$$

Resultado: sistema linear com $n-2$ equações e incógnitas $T_1, T_2, \ldots, T_{n-2}$

Próxima aula

Resumo de hoje:

- Resolvemos EDOs com condições iniciais (PVI) usando métodos como Euler, RK e `solve_ivp`.
- Discutimos rigidez e soluções implícitas.
- Estudamos problemas de valor de contorno (PVC), com destaque para o método do tiro.

Próxima aula:

- Retomaremos o método das diferenças finitas, agora aplicado a **Equações Diferenciais Parciais (EDPs)**.
- Veremos como discretizar o espaço e o tempo em problemas como condução de calor transiente.