EDOs

8 de fevereiro de 2022

1 Introdução

Uma **Equação Diferencial Ordinária (EDO)** relaciona uma função de uma **única** variável com as **derivadas** dessa função.

EDOs surgem constantemente em problemas de engenharia e ciência.

Exemplo 1. Pêndulo simples:

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\sin\theta = 0$$

$$\theta = \theta(t)$$

$$\theta(0) = \theta_0$$

$$\theta'(0) = \theta_0'$$

Note que $\theta' = d\theta/dt$.

Exemplo 2. Projétil em queda livre:

$$mv' = -mg - k|v|v$$

$$v = v(t)$$

$$v(0) = 0$$

Exemplo 3. Circuito elétrico:

$$\frac{di}{dt} = C\frac{d^2\epsilon}{dt^2} + \frac{1}{R}\frac{d\epsilon}{dt} + \frac{1}{L}\epsilon$$

$$i = i(t)\ , \epsilon = \epsilon(t)$$

$$i(0) = 0$$

i é a corrente, ϵ é a tensão aplicada, R é a resistência, L é a indutância e C é a capacitância.

2 Propriedades Gerais

Temos, como exemplo,

$$y' = \frac{dy}{dt} = f(t, y) \; , \qquad y(t_0) = y_0$$

Aqui, y é a variável dependente, e t é a variável independente.

Resolver a EDO significa encontrar a função $y(t)$ que satisfaça à **equação** e às **condições de contorno**, simultaneamente.

Ordem da EDO é a maior derivada presente na equação.

EDO geral:

$$a_n y^{(n)} + a_{n-1} y^{(n-1)} + \cdots + a_2 y'' + a_1 y' + a_0 y = f(t)$$

Nessa equação, $y^{(n)} = d^n y / dt^n$ e $a_i = a_i(t, y, y', y'', \cdots, y^{(n)})$ são os coeficientes.

EDO linear: equação na qual os coeficientes a_i não dependem de y nem de suas derivadas.

Alguns exemplos:

1. $y' + \alpha y = f(t)$ é uma EDO de primeira ordem linear, não homogênea, com coeficientes constantes
2. $y'' + \alpha t y = f(t)$ é uma EDO de segunda ordem linear, não homogênea, com coeficientes variáveis
3. $(y''')^2 + \alpha y' = 0$ é uma EDO de terceira ordem não linear, homogênea, com coeficientes variáveis
4. $y y'' + \alpha y = f(t)$ é uma EDO de segunda ordem não linear, não homogênea, com coeficientes variáveis

3 Tipos de Problemas

Temos dois tipos de problemas relacionados a EDOs.

Problemas de Valor Inicial ou de Propagação:

$$y' = f(t, y) \qquad y(t_0) = y_0 \qquad t_0 \leq t < \infty$$

Problemas de valor de contorno ou de equilíbrio (EDO com ordem 2 ou mais). Vamos considerar agora $y = y(x)$:

$$y'' + a_1(x, y, y')y' + a_2(x, y)y = f(x)$$

$$y(x_1) = y_1 \qquad y(x_2) = y_2 \qquad x_1 \leq x \leq x_2$$

Vamos focar aqui em problemas de valor inicial.

4 Método das Diferenças Finitas

Queremos resolver

$$y' = f(t, y) \ , \qquad y(0) = y_0$$

O objetivo do método das diferenças finitas é **transformar um problema de cálculo em um problema de álgebra**. Isso é feito em 4 etapas:

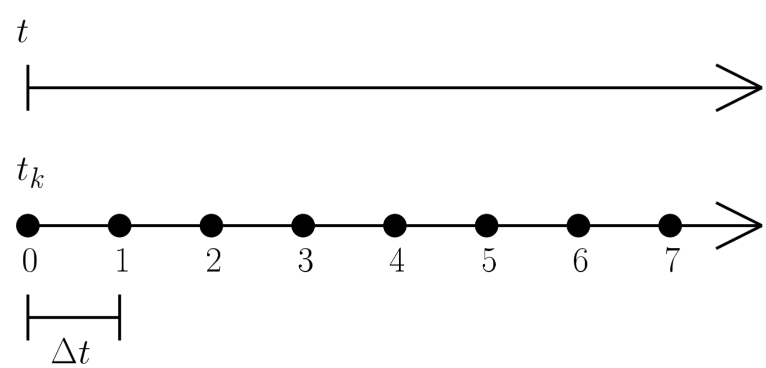
1. discretização do domínio
2. aproximação das derivadas usando série de Taylor
3. substituição das aproximações de volta na equação original, resultando na Equação de Diferenças Finitas (EDF)
4. resolução da EDF

Nesse método a EDO é transformada em várias EDFs.

Vamos dar uma olhada em cada uma dessas etapas.

4.1 Discretização (Etapa 1)

Vamos dividir o domínio em intervalos, resultando na malha de diferenças finitas.



Os pontos estão separados por Δt, que é chamado de **passo de tempo**. Vamos resolver a EDF apenas nesses pontos.

O primeiro ponto é o ponto 0, depois o 1 e assim por diante.

Assumindo que o domínio no tempo começa em $t = 0$, o tempo correspondente ao ponto 0 é $t_0 = 0$, ao ponto 1 é $t_1 = \Delta t$ e $t_2 = 2\Delta t$. De maneira geral:

$$t_k = k\Delta t$$

$$t_{k+1} = t_k + \Delta t$$

$$t_{k+2} = t_k + 2\Delta t$$

O valor da função y, que é a solução, no **tempo** k é escrito como

$$y(t_k) = y_k$$

De maneira similar, as derivadas serão representadas por

$$\frac{dy}{dt}(t_k) = y'(t_k) = y'\big|_k$$

4.2 Aproximação das Derivadas (Etapa 2)

Vamos representar a solução exata como sendo $\bar{y}(t)$ e a solução aproximada como $y(t)$.

Escrevendo uma série de Taylor para $\bar{y}$ em t_{k+1} em torno de t_k, temos

$$\bar{y}(t_{k+1}) = \bar{y}_{k+1} = \bar{y}_k + \bar{y}'\big|_k \Delta t + \frac{\bar{y}''\big|_{t=\tau}}{2}\Delta t^2$$

O último termo é o **resto**, com $t_k < \tau < t_{k+1}$. Isolando $\bar{y}'\big|_k$, resulta:

$$\bar{y}'\big|_k = \frac{\bar{y}_{k+1} - \bar{y}_k}{\Delta t} - \frac{\bar{y}''\big|_{t=\tau}}{2}\Delta t$$

Temos uma expressão exata para $\bar{y}'$ no tempo k que relaciona dois valores vizinhos de $\bar{y}$, que são $\bar{y}_k$ e $\bar{y}_{k+1}$.

Não conhecemos o último termo, mas sabemos que ele é proporcional a Δt. Ou seja, se $\bar{y}''$ for limitada, então esse último termo tende a zero quando Δt também tende a zero.

Vamos utilizar essa equação para o nosso **valor aproximado** y' (sem a barra), ignorando o último termo.

Assim:

$$y'\big|_k = \frac{y_{k+1} - y_k}{\Delta t}$$

Ao fazer essa aproximação estamos cometendo um **erro**. Não sabemos qual é o valor exato desse erro, mas sabemos que ele é proporcional a Δt. Dizemos que o erro é $O(\Delta t)$.

Esse erro na aproximação da derivada é chamado de **erro de truncamento**.

4.3 Substituição (Etapa 3)

EDO (Equação Diferencial Ordinária):

$$\bar{y}' = f(t, \bar{y}) \ , \qquad \bar{y}(0) = \bar{y}_0$$

EDF (Equação de Diferenças Finitas):

$$y'_k = \frac{y_{k+1} - y_k}{\Delta t} = f(t_k, y_k) = f_k \ , \qquad y_0 = \bar{y}_0$$

Ou, simplesmente:

$$\frac{y_{k+1} - y_k}{\Delta t} = f_k \ , \qquad k = 0, 1, 2, 3, \cdots \qquad y_0 = \bar{y}_0$$

Conclusão: transformamos uma EDO em um sistema de equações algébricas, as EDFs. Temos **uma equação para cada ponto** do domínio discretizado.

4.4 Resolução (Etapa 4)

Na equação acima, vamos isolar y_{k+1}, resultando:

$$y_{k+1} = y_k + \Delta t f_k \ , \qquad y_0 = \bar{y}_0$$

A equação acima é a equação que utilizaremos para resolver o problema numericamente. Esse é apenas um dos métodos disponíveis. Ele é chamado de **Método de Euler Explícito**, um método de diferenças finitas de primeira ordem.

4.5 Exemplo

Considere o seguinte problema:

$$\bar{y}' + \bar{y} = 0 \ , \qquad \bar{y}(0) = 1$$

Esse é o problema original. Comparando com o modelo apresentado acima, temos

$$f(t, \bar{y}) = -\bar{y}$$

Usando o Método de Euler Explícito, temos:

$$y_{k+1} = y_k + \Delta t f_k$$

$$y_{k+1} = y_k + \Delta t(-y_k)$$

$$y_{k+1} = y_k(1 - \Delta t)$$

Utilizando um passo de tempo $\Delta t = 0.1$, temos:

$$y_0 = 1 \quad \text{(condição inicial)}$$

$$y_1 = 1 \times (1 - 0.1) = 0.9$$

$$y_2 = 0.9 \times (1 - 0.1) = 0.81$$

$$y_3 = 0.81 \times (1 - 0.1) = 0.729$$

y_3 é uma aproximação para a solução verdadeira em $t = 3\Delta t$. A solução exata para o problema que estamos resolvendo é

$$\bar{y}(t) = e^{-t}$$

Assim, temos, para a solução exata e para a solução numérica, em $t = 0.3$:

$$y(0.3) = 0.729 \qquad \bar{y}(0.3) = 0.7408$$

O erro numérico para esse instante de tempo é obtido a partir da diferença entre os dois:

$$\text{erro} = |\bar{y} - y|$$

Espera-se que o erro diminua na medida em que Δt diminui. Porém, com um Δt menor são necessários mais passos.

4.6 Implementação em Python

Abaixo temos uma implementação da resolução acima em *python*.

[23]:
```
import numpy as np
import matplotlib.pyplot as plt

t = 0.0
t_final = 0.3
y = 1.0
dt = 0.1

while t < t_final-0.01*dt:
    y = y*(1.0-dt)     # essa é a parte fundamental
    t += dt
```

```
    print(f'Tempo = {t:3.2f}     y = {y:.5f}')

y_exata = lambda t : np.exp(-t)

erro = np.abs(y - y_exata(t_final))
print(f'\nErro = {erro:.5f}')
```

```
Tempo = 0.10     y = 0.90000
Tempo = 0.20     y = 0.81000
Tempo = 0.30     y = 0.72900

Erro = 0.01182
```

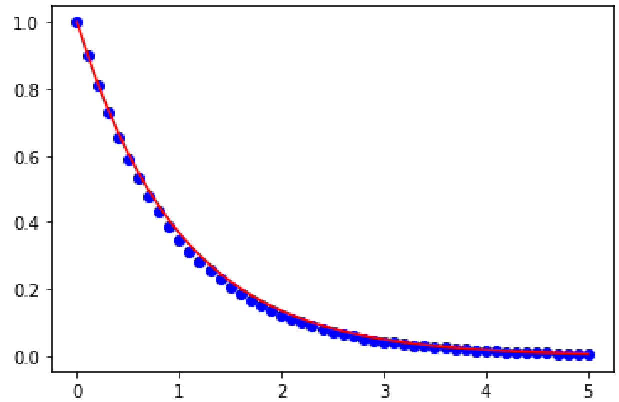
Em um dos exercícios do Trabalho é necessário fazer um gráfico do erro em função de Δt para um dado t.

No código acima estamos interessados apenas no valor final de y. Mas se quisermos o gráfico de $y \times t$, precisamos armazenar os valores intermediários. Veja o código abaixo.

[24]:
```
dt = 0.1     # esse é o passo de tempo
t_inicial = 0.0     # esse é o valor do tempo no início
t_final = 5.0     # esse é o valor final do tempo

# n é o número de passos de tempo
# pra dar certo com o tempo final, (t_final-t)/dt
# tem que ser inteiro
n = 1+int(round((t_final-t_inicial)/dt))

t = np.zeros(n,float)     # agora t é uma matriz
t[0] = t_inicial

y = np.zeros(n,float)     # agora y é uma matriz do numpy
y[0] = 1.0     # essa é a condição inicial

for k in range(n-1):
    t[k+1] = (k+1)*dt
    y[k+1] = y[k]*(1.0-dt)
    #print(f'Tempo = {t[k+1]:3.2f}     y = {y[k+1]:.5f}')

# poderíamos ter escrito o loop anterior
# deslocando k para a direita:
#for k in range(1,n):
#    t[k] = (k)*dt
#    y[k] = y[k-1]*(1.0-dt)
#    print(f'Tempo = {t[k]:3.2f}     y = {y[k]:.5f}')

# solução exata
y_exata = np.exp(-t)
```

```
# plotando o gráfico de y x t
plt.plot(t,y,'bo')
plt.plot(t,y_exata,'r-')
```

[24]: [<matplotlib.lines.Line2D at 0x7fdc1c46a130>]

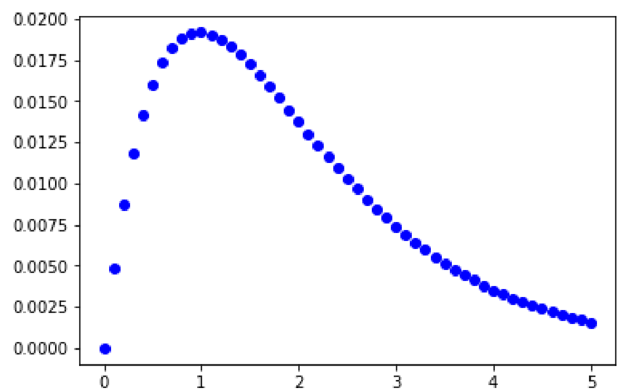


Geralmente plotamos os resultados numéricos como pontos e os resultados exatos (analíticos) como curvas contínuas.

Poderíamos fazer o gráfico do erro em função de t:

[25]:
```
erro = np.abs(y_exata-y)
plt.plot(t,erro,'bo')
```

[25]: [<matplotlib.lines.Line2D at 0x7fdc1c432610>]

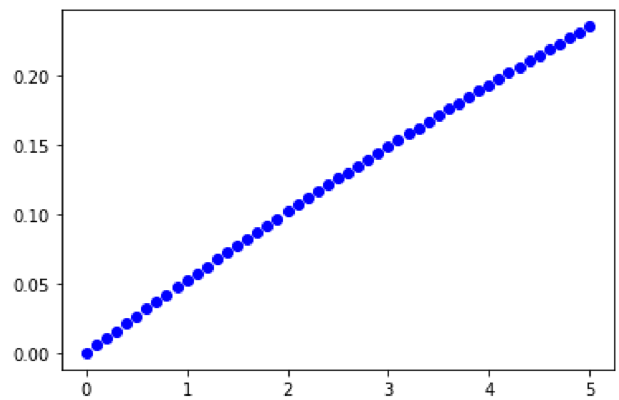


O erro diminui com o passar do tempo, o que não era esperado. Mas isso ocorre porque estamos olhando para o valor absoluto. Note que o valor de y diminui muito com o tempo, então é esperado que o erro diminua proporcionalmente.

Isso fica mais claro quando olhamos para o erro relativo:

[26]:
```
erro = np.abs((y_exata-y)/y_exata)
plt.plot(t,erro,'bo')
```

[26]: [<matplotlib.lines.Line2D at 0x7fdc1c384c40>]



4.7 Método de Euler Implícito

Para derivar o Método de Euler Implícito nós vamos aproximar $\bar{y}_k$ em torno de t_{k+1} usando um polinômio de Taylor:

$$\bar{y}(t_k) = \bar{y}_k = \bar{y}_{k+1} - \bar{y}'\big|_{k+1}\Delta t + \frac{\bar{y}''\big|_{t=\tau}}{2}\Delta t^2$$

O último termo é o **resto**, com $t_k < \tau < t_{k+1}$. Isolando $\bar{y}'\big|_{k+1}$, resulta:

$$\bar{y}'\big|_{k+1} = \frac{\bar{y}_{k+1} - \bar{y}_k}{\Delta t} - \frac{\bar{y}''\big|_{t=\tau}}{2}\Delta t$$

Agora temos uma expressão exata para $\bar{y}'$ no tempo $k+1$ que relaciona dois valores vizinhos de $\bar{y}$, que são $\bar{y}_k$ e $\bar{y}_{k+1}$.

Vamos utilizar essa equação para o nosso **valor aproximado** y', ignorando o último termo.

Assim:

$$y'\big|_{k+1} = \frac{y_{k+1} - y_k}{\Delta t}$$

Nessa aproximação estamos cometendo novamente um erro $O(\Delta t)$.

Substituindo essa aproximação no problema original, temos

$$y'\big|_{k+1} = \frac{y_{k+1} - y_k}{\Delta t} = f_{k+1} = f(t_{k+1}, y_{k+1})$$

Ou:

$$\frac{y_{k+1} - y_k}{\Delta t} = f_{k+1}$$

Finalmente, resolvendo para y_{k+1}, temos

$$y_{k+1} = y_k + \Delta t f_{k+1}$$

Esse é o **Método de Euler Implícito**. Trata-se de um método de diferenças de primeira ordem.

A diferença aqui é que agora f é calculada em $k+1$ e não em k. Portanto, no lado direito dessa última equação pode haver termos em função de y_{k+1}, o que torna o método um pouco mais complicado em alguns casos.

4.8 Exemplo

Considere o mesmo problema do exemplo anterior:

$$\bar{y}' + \bar{y} = 0 \ , \qquad \bar{y}(0) = 1$$

Comparando com o modelo apresentado acima, temos

$$f(t, \bar{y}) = -\bar{y}$$

Usando o Método de Euler Implícito, temos a seguinte EDF:

$$\frac{y_{k+1} - y_k}{\Delta t} = f_{k+1}$$

ou

$$y_{k+1} = y_k + \Delta t f_{k+1}$$

Substituindo o valor de f_{k+1}:

$$y_{k+1} = y_k + \Delta t(-y_{k+1})$$

Resolvendo para y_{k+1}:

$$y_{k+1} = \frac{y_k}{1 + \Delta t} \ , \qquad \text{com } y_0 = 1$$

Utilizando um passo de tempo $\Delta t = 0.1$, temos:

$$y_0 = 1 \quad \text{(condição inicial)}$$

$$y_1 = \frac{1}{1+0.1} = 0.909$$

$$y_2 = \frac{0.909}{1+0.1} = 0.826$$

$$y_3 = \frac{0.826}{1+0.1} = 0.751$$

Assim, temos, para a solução exata ($\bar{y} = e^{-t}$) e para a solução numérica, em $t = 0.3$:

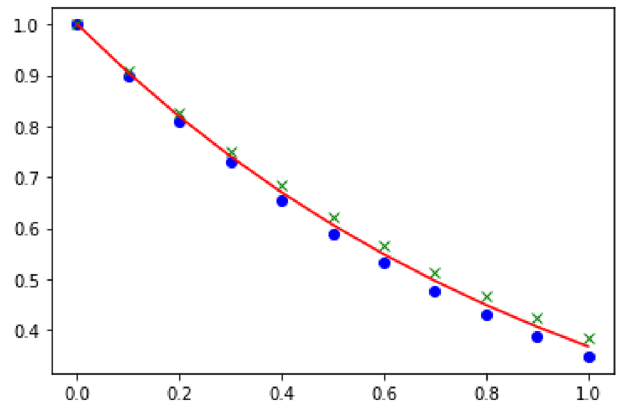
$$y(0.3) = 0.751 \qquad \bar{y}(0.3) = 0.7408$$

No código abaixo temos a implementação do método implícito e comparação dos seus resultados com aqueles do método explícito e da solução exata.

[28]:
```
dt = 0.1    # esse é o passo de tempo
t_inicial = 0.0    # esse é o valor do tempo no início
t_final = 1.0    # esse é o valor final do tempo

# n é o número de passos de tempo
# pra dar certo com o tempo final, (t_final-t)/dt
# tem que ser inteiro
n = 1+int(round((t_final-t_inicial)/dt))

t = np.zeros(n,float)    # agora t é uma matriz
t[0] = t_inicial

y_exp = np.zeros(n,float)    # método explícito
y_exp[0] = 1.0    # essa é a condição inicial

y_imp = np.zeros(n,float)    # método implícito
y_imp[0] = 1.0    # essa é a condição inicial

for k in range(n-1):
    t[k+1] = (k+1)*dt
    y_exp[k+1] = y_exp[k]*(1.0-dt)
    y_imp[k+1] = y_imp[k]/(1.0+dt)

    #print(f'Tempo = {t[k+1]:3.2f}    y_exp = {y_exp[k+1]:.5f}    y_imp =␣
 ↪{y_imp[k+1]:.5f}')
# solução exata
y_exata = np.exp(-t)
```

```
# plotando o gráfico de y x t
plt.plot(t,y_exp,'bo')
plt.plot(t,y_imp,'gx')
plt.plot(t,y_exata,'r-')
```

[28]: [<matplotlib.lines.Line2D at 0x7fdc1c2cc130>]

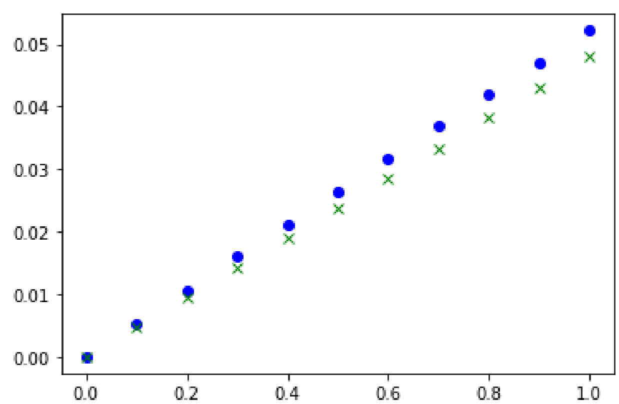


A solução dada pelo método implícito está acima da solução dada pelo método explícito. Ainda, o método implícito parece ser um pouco mais preciso.

Vamos dar uma olhada nos erros.

[29]:
```
erro_exp = np.abs((y_exata-y_exp)/y_exata)
erro_imp = np.abs((y_exata-y_imp)/y_exata)
plt.plot(t,erro_exp,'bo')
plt.plot(t,erro_imp,'gx')
```

[29]: [<matplotlib.lines.Line2D at 0x7fdc1c657a00>]



Vamos ver agora o que acontece quando a gente vai aumentando o valor de dt.

Primeiro no **Método de Euler Explícito**:

[46]:
```
t = 0.0
t_final = 1000.0
y = 1.0
dt = 2.1

while t < t_final-0.01*dt:
    y = y*(1.0-dt)
    t += dt

print(f'Tempo = {t:3.1f}    y = {y:.5f}')

y_exata = lambda t : np.exp(-t)

erro = np.abs(y - y_exata(t_final))
print(f'\nErro = {erro:.5f}')
```

```
Tempo = 1001.7    y = -55502279543799513088.00000

Erro = 55502279543799513088.00000
```

Agora com o **Método de Euler Implícito**:

[40]:
```
t = 0.0
t_final = 100.0
y = 1.0
dt = 10.0

while t < t_final-0.01*dt:
    y = y/(1.0+dt)    # essa é a parte fundamental
    t += dt

print(f'Tempo = {t:3.1f}    y = {y:.5f}')

y_exata = lambda t : np.exp(-t)

erro = np.abs(y - y_exata(t_final))
print(f'\nErro = {erro:.8f}')
```

```
Tempo = 100.0    y = 0.00000

Erro = 0.00000000
```

O Método Explícito 'explode' quando o Δt é grande, mas o Implícito não.

Vamos ver o porquê na próxima seção.

5 Consistência, Ordem, Estabilidade e Convergência

Vamos dar uma olhada em alguns conceitos importantes com relação à solução numérica de EDOs.

Só pra relembrar: **EDO** é Equação Diferencial Ordinária e **EDF** é Equação de Diferenças Finitas

Uma EDF é **consistente** com uma EDO se a diferença entre elas se anula quando $\Delta t \to 0$. Em outras plavras, a EDF aproxima a EDO.

A **ordem** de uma EDF é a taxa com que o erro global decresce na medida em que Δt se aproxima de zero.

Uma EDF é **estável** se produz uma solução delimitada (*bounded*) para uma EDO estável e é **instável** de produz uma solução não delimitada (*unbounded*) para uma EDO estável.

Um método de diferenças finitas é **convergente** se a solução numérica da EDF (isto é, os valores numéricos) aproxima a solução exata da EDO na medida em que $\Delta t \to 0$.

5.1 Consistência

Transforme a EDF de volta em um EDO e veja se as duas se igualam quando $\Delta t \to 0$.

Exemplo. considere a EDO:

$$\bar{y}' = -\alpha \bar{y}$$

Usando o Método de Euler Explícito, temos a seguinte EDF:

$$\frac{y_{k+1} - y_k}{\Delta t} = -\alpha y_k$$

Mas

$$y_{k+1} = y_k + \Delta t\, y'\big|_k + \frac{1}{2}(\Delta t)^2 y''\big|_k + \frac{1}{6}(\Delta t)^3 y'''\big|_k + \cdots$$

Substituindo essa valor de volta na EDF:

$$y'\big|_k = -\alpha y_k - \frac{1}{2}(\Delta t) y''\big|_k - \frac{1}{6}(\Delta t)^2 y'''\big|_k + \cdots$$

Essa equação é chamada de **Equação Diferencial Modificada** (EDM).

Fazendo $\Delta t \to 0$ na EDM, recuperamos a EDO original.

Isso significa que a EDF é consistente com a EDO.

5.2 Ordem

A ordem da EDF é o menor expoente de Δt que aparece na EDM.

No caso do exemplo acima, a odem da EDF é Δt. Ou seja, a EDF é uma aproximação $O(\Delta t)$ da EDO.

5.3 Estabilidade

Toda EDF deve ser analisada com relação à estabilidade.

EDO:

$$\bar{y}' = f(t, \bar{y})$$

EDF:

$$y' = f(t, y)$$

A EDF pode ser escrita como:

$$y_{k+1} = G y_k$$

G é chamado de **Fator de Amplificação** da EDF.

Temos, portanto:

$$y_1 = G y_0 \qquad y_2 = G y_1 = G^2 y_0$$

$$y_3 = G y_2 = G^3 y_0 \qquad \cdots$$

$$y_n = G^n y_0$$

Para que y_n seja limitado e seu valor não 'exploda' quando $n \to \infty$, devemos ter

$$|G| \leq 1$$

5.3.1 Exemplo

Para fazer a análise de estabilidade, vamos considerar a seguinte equação teste:

$$\bar{y}' = -\alpha \bar{y} \ , \qquad \alpha > 0$$

Primeiramente, vamos analisar o **Método de Euler Explícito**.

Temos:

$$\frac{y_{k+1} - y_k}{\Delta t} = f_k = -\alpha y_k$$

$$y_{k+1} = y_k - \Delta t \alpha y_k = (1 - \alpha \Delta t) y_k$$

Assim:

$$y_{k+1} = (1 - \alpha\Delta t) y_k$$

O fator de amplificação dessa EDF é dado por

$$G = 1 - \alpha\Delta t$$

Para que a EDF seja estável, devemos ter

$$|G| \leq 1$$

$$-1 \leq G \leq 1$$

$$-1 \leq 1 - \alpha\Delta t \leq 1$$

$$\Delta t \geq 0 \qquad \Delta t \leq \frac{2}{\alpha}$$

Por definição devemos ter $\Delta t > 0$, então a condição importante nesse caso é a segunda.

Para que o Método de Euler Explícito seja estável devemos ter $\Delta t \leq 2/\alpha$. O Método de Euler Explícito é **condicionalmente estável**.

Agora vamos ver o que acontece com o **Método de Euler Implícito**.

Temos:

$$\frac{y_{k+1} - y_k}{\Delta t} = f_{k+1} = -\alpha y_{k+1}$$

$$y_{k+1} + \Delta t \alpha y_{k+1} = y_k$$

Assim:

$$y_{k+1} = y_k \left(\frac{1}{1 + \alpha\Delta t} \right)$$

O fator de amplificação dessa EDF é dado por

$$G = \frac{1}{1 + \alpha\Delta t}$$

Como $\Delta t > 0$, $|G| \leq 1$ sempre. Consequentemente, o método de Euler implícito é **incondicionalmente estável**.

Conclusão: o método implícito vai ser estável sempre. Se você usar um Δt grande ele vai dar uma resposta ruim, mas nunca vai 'explodir'.

5.4 Convergência

A convergência de um método de diferenças finitas é **garantida** demonstrando-se que a EDF é **consistente** e **estável**.

6 Alguns Métodos para Resolver EDOs Numericamente

EDO:

$$y' = f(t, y)\ , \qquad y(t_0) = y_0$$

6.1 Euler Explícito (Primeira Ordem)

$$y_{k+1} = y_k + \Delta t f_k$$

6.2 Euler Implícito (Primeira Ordem)

$$y_{k+1} = y_k + \Delta t f_{k+1}$$

6.3 Método do Ponto Médio Modificado (Segunda Ordem)

O método envolve duas etapas:

$$y^p_{k+1/2} = y_k + \frac{\Delta t}{2} f_k$$

$$y_{k+1} = y_k + \Delta t f^p_{k+1/2}$$

Aqui, temos

$$f^p_{k+1/2} = f(t_{k+1/2}, y^p_{k+1/2})$$

$$t_{k+1/2} = t_k + \frac{\Delta t}{2}$$

6.4 Método do Trapézio Modificado (Segunda Ordem)

O método envolve duas etapas:

$$y^p_{k+1} = y_k + \Delta t f_k$$

$$y_{k+1} = y_k + \frac{\Delta t}{2}(f_k + f^p_{k+1})$$

Aqui, temos

$$f^p_{k+1} = f(t_{k+1}, y^p_{k+1})$$

6.5 Método de Runge-Kutta de Segunda Ordem

$$y_{k+1} = y_k + \frac{1}{2}(k_1 + k_2)$$

$$k_1 = \Delta t f(t_k, y_k)$$

$$k_2 = \Delta t f(t_k + \Delta t, y_k + k_1)$$

6.6 Método de Runge-Kutta de Quarta Ordem

$$y_{k+1} = y_k + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = \Delta t f(t_k, y_k)$$

$$k_2 = \Delta t f(t_k + \frac{\Delta t}{2}, y_k + \frac{k_1}{2})$$

$$k_3 = \Delta t f(t_k + \frac{\Delta t}{2}, y_k + \frac{k_2}{2})$$

$$k_4 = \Delta t f(t_k + \Delta t, y_k + k_3)$$

Comentário: todos esses métodos são chamados de **Métodos de Passo Simples**, pois usam apenas y_k para encontrar y_{k+1}. Existem ainda os **Métodos de Passos Múltiplos**, que utilizam os valores de y_k e pelo menos y_{k-1}, podendo ainda usar os valores de y_{k-2}, y_{k-3} $\cdots$. Não vamos estudar métodos de passos múltiplos aqui em nosso curso.