

Universidade Federal do Paraná

# EDO - Método de Runge-Kutta

Discentes: Grazielle Ferreira, Natália Freire e Thaynara Ramos

Docente: Prof. Dr. Rogério Vargas



Junho de 2025

- 1 Introdução
- 2 Método de Runge-Kutta
- 3 Ordem
  - 3.1 RK2
  - 3.2 RK4
- 4 Comparação com o Método Euler
- 5 Aplicações do Método de Runge-Kutta
- 6 Referências

# Introdução

## O que é uma EDO?

**Equação Diferencial Ordinária** - **EDO** - é uma equação que relaciona uma função incógnita  $y$  com suas respectivas derivadas em relação a uma única variável independente. Essas equações modelam fenômenos em diversas áreas da engenharia e ciências naturais.

Segue o exemplo:

$$\frac{dy}{dx} = x + y$$

onde  $y$  é a função incógnita, e a equação relaciona  $y$  e sua derivada em relação a  $x$ .

Existem diversos métodos para resolver EDOs e entre eles está o **Método de Runge-Kutta**, que será abordado nessa apresentação.

O **Método de Runge-Kutta** foi desenvolvido pelos matemáticos alemães **Carl David Runge** (1856-1927) e **M. Wilhelm Kutta** (1867-1944).

O interesse em resolver numericamente equações diferenciais levou à formulação do Método de Runge-Kutta. Em 1901, Kutta publicou um artigo propondo o método, que mais tarde foi aprimorado por Runge.

Hoje em dia o método é muito utilizado em softwares para a solução de problemas devido a sua precisão e eficiência.

# Método de Runge-Kutta

# Método de Runge-Kutta

O **Método de Runge-Kutta** é um dos métodos mais utilizados para a solução de EDOs. Ele é dividido por ordens, ou seja, **RK1**, **RK2**, **RK3** e **RK4**, onde quanto maior a ordem, maior a precisão.

O método de quarta ordem - **RK4** - é o mais utilizado para obter soluções aproximadas de um valor inicial.

Diferente do método de Taylor, que exige o cálculo de derivadas de ordem superior, o método de Runge-Kutta encontra aproximações precisas utilizando apenas a função original.

# Polinômio de Taylor

Pelo **Polinômio de Taylor** com termo de resto, se uma função  $y(x)$  tiver  $k + 1$  derivadas contínuas em um intervalo, então:

$$y(x) = y(a) + y'(a) \frac{xa}{1!} + \dots + y^{(k)}(a) \frac{(xa)^k}{k!} + y^{(k+1)}(c) \frac{(xa)^{(k+1)}}{(k+1)!}$$

Porém, há uma limitação. Para ordens superiores, o polinômio exige derivadas de alta ordem. O Método de Runge-Kutta pode ser utilizado para contornar essa situação.



# Método Runge-Kutta

Tendo em vista o que foi apresentado até agora, entendemos então que o Método de Runge-Kutta é uma técnica numérica **importante para resolver equações diferenciais ordinárias** já que possui:

- Precisão;
- Praticidade;
- Estabilidade.

Ele é **especialmente útil quando é difícil ou impossível encontrar a solução exata analiticamente.**

Ordem

O Método de Runge-Kutta possui:

- 1 Runge-Kutta de Primeira Ordem - RK1
- 2 Runge-Kutta de Segunda Ordem - RK2
- 3 Runge-Kutta de Terceira Ordem - RK3
- 4 Runge-Kutta de Quarta Ordem - RK4

A ordem do método indica o número de pontos usado em um subintervalo para determinar o valor da **inclinação** - uma constante que é obtida através do cálculo da inclinação em vários pontos no interior do subintervalo.

No Método Runge-Kutta de **Segunda Ordem - RK2**, também conhecido como **Método de Heun**, temos a aproximação da solução usando a média da inclinação no início e no fim do passo.

As fórmulas para a utilização desse método são:

$$k_1 = f(x_n, y_n)$$

$$k_2 = f(x_n + h, y_n + h.k_1)$$

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

No Método Runge-Kutta de **Quarta Ordem - RK4** utilizamos quatro estimativas de inclinação para obter maior precisão dos resultados, sendo elas:

$$\begin{aligned}k_1 &= f(x_n, y_n) & k_3 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right) \\k_2 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) & k_4 &= f(x_n + h, y_n + h.k_3) \\y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)\end{aligned}$$

Possui **erro global** de ordem  $O(h^4)$ , o que traz mais precisão.

# Comparação com o Método Euler

# Comparação com o Método Euler

Método	Ordem de erro	Precisão	Estabilidade	Custo Computacional
Euler	1	Baixa	Ruim	Baixo
RK2	2	Média	Boa	Médio
RK4	4	Alta	Excelente	Alto

Figure: Comparação entre os métodos Euler, RK2 e RK4.

- **Euler**: simples, mas impreciso e instável.
- **RK2**: mais exato, porém com um pouco mais de cálculos.
- **RK4**: alta precisão, ideal para aplicações reais na engenharia.

# Aplicações do Método de Runge-Kutta



Relembrando as fórmulas do **RK4** e do **RK2**,  
respectivamente:

$$k_1 = f(x_n, y_n)$$

$$k_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right)$$

$$k_3 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right)$$

$$k_4 = f(x_n + h, y_n + h.k_3)$$

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

$$k_1 = f(x_n, y_n)$$

$$k_2 = f(x_n + h, y_n + h.k_1)$$

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

Apresentando os significados das incógnitas:

$f$  : a função que define a EDO;

$x_n$  : o valor inicial de  $x$ ;

$y_n$  : o valor inicial de  $y$  em  $x_n$ ;

$h$  : o tamanho do passo;

$y_{n+1}$  : valor atualizado de  $y$ .

- **Exercício:**

A função abaixo representa a taxa de recalque do solo no adensamento de uma fundação:

$$\frac{ds}{dt} = 0,8s + 2t$$

onde,  $s_0 = 0\text{mm}$  e  $t$  vai de 0 até 2h, com  $h = 1\text{h}$ .

Utilize o **Método de Runge-Kutta - RK4** - para calcular essa taxa.

Antes de realizar os cálculos, é importante "transformar" as fórmulas com as suas incógnitas para evitar confusões.

$$k_1 = f(x_n, y_n)$$

$$k_1 = f(t_n, s_n)$$

$$k_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right)$$

$$k_2 = f\left(t_n + \frac{h}{2}, s_n + \frac{h}{2}k_1\right)$$

$$k_3 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right)$$

$$k_3 = f\left(t_n + \frac{h}{2}, s_n + \frac{h}{2}k_2\right)$$

$$k_4 = f(x_n + h, y_n + h.k_3)$$

$$k_4 = f(t_n + h, s_n + h.k_3)$$

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

O primeiro passo que deve ser realizado para dar início aos cálculos é o **Cálculo da Quantidade de Passos**.

Ela é calculada através de:

$$n = \frac{t_f - t_0}{h}$$

Nesse caso, como **t** vai de 0 até 2h e **h** é igual a 1h, temos que:

$$n = \frac{2 - 0}{1} = 2$$

Dessa forma, entende-se que os cálculos do Método de Runge-Kutta - RK4 - serão realizados 2 vezes.

# Aplicações - Exercício

Abaixo seguem os resultados obtidos através dos cálculos com o método RK4 e na sequência com o método RK2 para realizar uma comparação entre os métodos.

Passo	$t_n$	$s_n$	$k_1$	$k_2$	$k_3$	$k_4$	$s_{n+1}$
1	0	0	0	1	1,4	3,12	1,32
2	1	1,32	3,056	5,2784	6,1674	9,9899	7,3096

Table: RK4

Passo	$t_n$	$s_n$	$k_1$	$k_2$	$s_{n+1}$
1	0	0	0	2	1
2	1	1	2,8	7,04	5,92

Table: RK2

# Aplicações - Python - RK4

```
1 # RK4
2
3 def rk4(f, t0, y0, h, num_steps):
4
5     # f: A função que define a EDO, no formato f(t, y) = dy/dt.
6     t0: O valor inicial de t.
7     y0: O valor inicial de y em t0.
8     h: O tamanho do passo.
9     num_steps: O número de passos a serem dados.
10
11     t_values = [t0]
12     y_values = [y0]
13
14     for i in range(num_steps):
15         t = t_values[-1]
16         y = y_values[-1]
17
18         # Passo 1: Calcular k1
19         k1 = h * f(t, y)
20
21         # Passo 2: Calcular k2
22         k2 = h * f(t + h/2, y + k1/2)
23
24         # Passo 3: Calcular k3
25         k3 = h * f(t + h/2, y + k2/2)
26
27         # Passo 4: Calcular k4
28         k4 = h * f(t + h, y + k3)
29
30         # Calcular a nova aproximação de y
31         y_next = y + (k1 + 2*k2 + 2*k3 + k4) / 6
32
33         # Atualizar os valores de t e y
34         t_values.append(t + h)
35         y_values.append(y_next)
36
37     return list(zip(t_values, y_values))
38
39 # Definindo a função f(t, y) para a EDO dy/dt = t + y
40 def f(t, y):
41     return t + y
```

```
43 t0 = float(input("Digite o valor inicial de t (t0): "))
44 y0 = float(input("Digite o valor inicial de y em t0 (y0): "))
45 h = float(input("Digite o tamanho do passo (h): "))
46 tf = float(input("Digite o valor final de t (tf): "))
47
48 # Calcular o número de passos com base em t0, tf e h
49 num_steps = int((tf - t0) / h)
50
51 # Resolvendo a EDO usando Runge-Kutta de 4ª ordem
52 solution = rk4(f, t0, y0, h, num_steps)
53
54 # Imprimindo a solução
55 for t, y in solution:
56     print(f"t = {t:.2f}, y = {y:.5f}")
57
58
59 # Criando um Gráfico
60 import matplotlib.pyplot as plt
61
62 # Extraíndo as listas de t e y da solução
63 t_values_rk4 = [point[0] for point in solution]
64 y_values_rk4 = [point[1] for point in solution]
65
66 # Criando gráfico de linha para o RK4
67 plt.plot(t_values_rk4, y_values_rk4, marker='o', linestyle='-',
68         color='blue', label='RK4')
69
70 # Adicionando título e rótulos
71 plt.title("Solução da EDO usando RK4")
72 plt.xlabel('t')
73 plt.ylabel('y')
74
75 # Adicionando legenda
76 plt.legend()
77
78 # Exibindo gráfico
79 plt.grid(True)
80 plt.show()
```

Figure: Python parte 1.

Figure: Python parte 2.

## Resultados:

```
Digite o valor inicial de t (t0): 0
Digite o valor inicial de y em t0 (y0): 1
Digite o tamanho do passo (h): 0.1
Digite o valor final de t (tf): 2
t = 0.00, y ≈ 1.00000
t = 0.10, y ≈ 1.11034
t = 0.20, y ≈ 1.24281
t = 0.30, y ≈ 1.39972
t = 0.40, y ≈ 1.58365
t = 0.50, y ≈ 1.79744
t = 0.60, y ≈ 2.04424
t = 0.70, y ≈ 2.32750
t = 0.80, y ≈ 2.65108
t = 0.90, y ≈ 3.01920
t = 1.00, y ≈ 3.43656
t = 1.10, y ≈ 3.90833
t = 1.20, y ≈ 4.44023
t = 1.30, y ≈ 5.03859
t = 1.40, y ≈ 5.71039
t = 1.50, y ≈ 6.46337
t = 1.60, y ≈ 7.30605
t = 1.70, y ≈ 8.24788
t = 1.80, y ≈ 9.29928
t = 1.90, y ≈ 10.47177
t = 2.00, y ≈ 11.77809
```

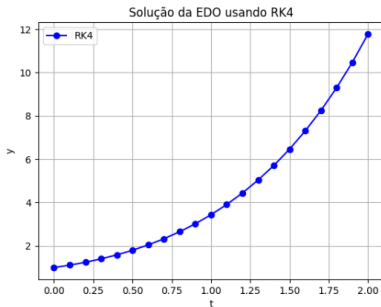


Figure: Resultado do Gráfico.

Figure: Resultado dos Cálculos.



# Aplicações - Python - RK2

```
1 # RK2
2
3 def rk2(f, t0, y0, h, num_steps):
4
5     # f: A função que define a EDO, no formato f(t, y) = dy/dt.
6     # t0: O valor inicial de t.
7     # y0: O valor inicial de y em t0.
8     # h: O tamanho do passo.
9     # num_steps: O número de passos a serem dados.
10
11     t_values = [t0]
12     y_values = [y0]
13
14     for i in range(num_steps):
15         t = t_values[-1]
16         y = y_values[-1]
17
18         # Passo 1: Calcular k1
19         k1 = h * f(t, y)
20
21         # Passo 2: Calcular k2
22         k2 = h * f(t + h, y + k1)
23
24         # Calcular a nova aproximação de y
25         y_next = y + (k1 + k2)/2
26
27         # Atualizar os valores de t e y
28         t_values.append(t + h)
29         y_values.append(y_next)
30
31     return list(zip(t_values, y_values))
32
33 # Definindo a função f(t, y) para a EDO dy/dt = t + y
34 def f(t, y):
35     return y - t**2 + 1
36
37 t0 = float(input("Digite o valor inicial de t (t0): "))
38 y0 = float(input("Digite o valor inicial de y em t0 (y0): "))
39 h = float(input("Digite o tamanho do passo (h): "))
40 tf = float(input("Digite o valor final de t (tf): "))
```

Figure: Python parte 1.

```
42 # Calcular o número de passos com base em t0, tf e h
43 num_steps = int((tf - t0) / h)
44
45 # Resolvendo a EDO usando Runge-Kutta de 2ª ordem
46 solution = rk2(f, t0, y0, h, num_steps)
47
48 # Imprimindo a solução
49 for t, y in solution:
50     print(f"t = {t:.4f}, y = {y:.6f}")
51
52
53 # prompt: criar um gráfico pro método rk2
54 import matplotlib.pyplot as plt
55
56 # Extraíndo as listas de t e y da solução
57 t_values_rk2 = [point[0] for point in solution]
58 y_values_rk2 = [point[1] for point in solution]
59
60 # Criando gráfico de linha para o RK2
61 plt.plot(t_values_rk2, y_values_rk2, marker='o', linestyle='-',
62          color='blue', label='RK2')
63
64 # Adicionando título e rótulos
65 plt.title('Solução da EDO usando RK2')
66 plt.xlabel('t')
67 plt.ylabel('y')
68
69 # Adicionando legenda
70 plt.legend()
71
72 # Exibindo gráfico
73 plt.grid(True)
74 plt.show()
```

Figure: Python parte 2.

## Resultados:

```
Digite o valor inicial de t (t0): 0
Digite o valor inicial de y em t0 (y0): 1
Digite o tamanho do passo (h): 0.1
Digite o valor final de t (tf): 2
t = 0.0000, y ≈ 1.000000
t = 0.1000, y ≈ 1.209500
t = 0.2000, y ≈ 1.438948
t = 0.3000, y ≈ 1.688337
t = 0.4000, y ≈ 1.957662
t = 0.5000, y ≈ 2.246917
t = 0.6000, y ≈ 2.556093
t = 0.7000, y ≈ 2.885183
t = 0.8000, y ≈ 3.234177
t = 0.9000, y ≈ 3.603066
t = 1.0000, y ≈ 3.991838
t = 1.1000, y ≈ 4.400481
t = 1.2000, y ≈ 4.828981
t = 1.3000, y ≈ 5.277324
t = 1.4000, y ≈ 5.745493
t = 1.5000, y ≈ 6.233470
t = 1.6000, y ≈ 6.741234
t = 1.7000, y ≈ 7.268764
t = 1.8000, y ≈ 7.816034
t = 1.9000, y ≈ 8.383018
t = 2.0000, y ≈ 8.969685
```

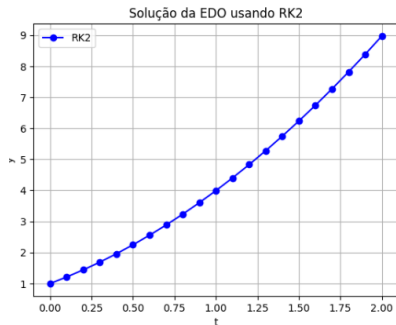


Figure: Resultado do Gráfico.

Figure: Resultado dos Cálculos.

# Aplicações - Planilha - RK4

EDO's - Método Runge-Kutta - RK4			f	t0	tf	y0	h	n		
Fórmulas			t + y	0	2	1	0,1	20		
$\frac{dy}{dt} = f(t,y)$										
$y(t_0) = y_0$										
$k_1 = h \cdot f(t_n, y_n)$			Passo	tn	yn	k1	k2	k3	k4	yn+1
			0	0,0	1,000000	1,000000	1,100000	1,105000	1,210500	1,110342
			1	0,1	1,110342	1,210342	1,320859	1,326385	1,442980	1,242805
$k_2 = h \cdot f(t_n + \frac{h}{2}, y_n + \frac{k_1}{2})$			2	0,2	1,242805	1,442805	1,564945	1,571052	1,699910	1,399717
$k_3 = h \cdot f(t_n + \frac{h}{2}, y_n + \frac{k_2}{2})$			3	0,3	1,399717	1,699717	1,834703	1,841452	1,983862	1,583648
$k_4 = h \cdot f(t_n + h, y_n + k_3)$			4	0,4	1,583648	1,983648	2,132831	2,140290	2,297677	1,797441
			5	0,5	1,797441	2,297441	2,462313	2,470557	2,644497	2,044236
$y_{n+1} = y_n + \frac{h}{6} (k_1 + 2k_2 + 2k_3 + k_4)$			6	0,6	2,044236	2,644236	2,826448	2,835558	3,027792	2,327503
$n = \frac{t_f - t_0}{h}$			7	0,7	2,327503253	3,027503253	3,228878416	3,238947174	3,451397971	2,651079127
			8	0,8	2,651079127	3,451079127	3,673633083	3,684760781	3,919555205	3,019202828
			9	0,9	3,019202828	3,919202828	4,165162969	4,177460976	4,436948925	3,436559488
f: A função que define a EDO, no formato f(t, y) = dy/dt.			10	1	3,436559488	4,436559488	4,708387463	4,721978861	5,008757374	3,90832698
t0: O valor inicial de t.			11	1,1	3,90832698	5,00832698	5,308743329	5,323764147	5,640703395	4,440227736
tf: O valor final de t.			12	1,2	4,440227736	5,640227736	5,972239122	5,988839692	6,339111705	5,03858602
y0: O valor inicial de y em t0.			13	1,3	5,03858602	6,33858602	6,705515321	6,723861786	7,110972199	5,710391227
h: O tamanho do passo.			14	1,4	5,710391227	7,110391227	7,515910789	7,536186767	7,964009904	6,463367831
n: O número de passos a serem dados.			15	1,5	6,463367831	7,963367831	8,411536223	8,433944642	8,906762296	7,306052696
			16	1,6	7,306052696	8,906052696	9,40135533	9,426120462	9,948664742	8,247880513
			17	1,7	8,247880513	9,947880513	10,49527454	10,52264424	11,10014494	9,299278229
			18	1,8	9,299278229	11,09927823	11,70424214	11,73449034	12,37272726	10,4717694
			19	1,9	10,4717694	12,3717694	13,04035787	13,0737873	13,77914813	11,77808953
			20	2	11,77808953	13,77808953	14,51699401	14,55393924	15,33348346	13,23231353

Figure: Planilha do Método RK4.

# Aplicações - Planilha - RK2

EDO's - Método Runge-Kutta - RK2			f	t0	tf	y0	h	n
Fórmulas			t + y	0	2	1	0,1	20
$\frac{dy}{dt} = f(t, y)$ $y(t_0) = y_0$ $k_1 = h \cdot f(t_n, y_n)$ $k_2 = h \cdot f(t_n + h, y_n + k_1)$ $y_{n+1} = y_n + \frac{h}{2} (k_1 + k_2)$ $n = \frac{t_f - t_0}{h}$			Passo	tn	yn	k1	k2	yn+1
			0	0,0	1,000000	1,000000	1,200000	1,110000
			1	0,1	1,110000	1,210000	1,431000	1,242050
			2	0,2	1,242050	1,442050	1,686255	1,398465
			3	0,3	1,398465	1,698465	1,968312	1,581804
			4	0,4	1,581804	1,981804	2,279985	1,794894
			5	0,5	1,794894	2,294894	2,624383	2,040857
			6	0,6	2,040857	2,640857	3,004943	2,323147
			7	0,7	2,323147	3,023147	3,425462	2,645578
			8	0,8	2,645578	3,445578	3,890136	3,012364
			9	0,9	3,012364	3,912364	4,403600	3,428162
			10	1,0	3,428162	4,428162	4,970978	3,898119
			11	1,1	3,898119	4,998119	5,597931	4,427921
			12	1,2	4,427921	5,627921	6,290713	5,023853
			13	1,3	5,023853	6,323853	7,056238	5,692857
			14	1,4	5,692857	7,092857	7,902143	6,442607
			15	1,5	6,442607426	7,942607426	8,836868169	7,281581206
			16	1,6	7,281581206	8,881581206	9,869739327	8,219147233
			17	1,7	8,219147233	9,919147233	11,01106196	9,265657692
			18	1,8	9,265657692	11,06565769	12,27222346	10,43255175
			19	1,9	10,43255175	12,33255175	13,66580693	11,73246968
			20	2	11,73246968	13,73246968	15,20571665	13,179379

Figure: Planilha do Método RK2.

# Referências

VALLE, Karine Nayara F.. Métodos Numéricos de Euler e Runge-Kutta. Disponível em:  
<http://hdl.handle.net/1843/48530>

MOTTA, Matheus C.; SILVA, Rosana M. da. Métodos de Runge-Kutta. Disponível em:  
<https://www.dme.ufcg.edu.br/6semanamat/resumos/a14.pdf>.  
Acesso em: 08 jun. 2025.

STERZA, Rafael de Lima; BRANDI, Analice Costacurta. Comparação entre métodos numéricos: Runge-Kutta de quarta ordem e previsor-corretor. Disponível em:  
<https://www.fc.unesp.br/Home/Departamentos/Matematica/revista-comparacao-entre-metodos-numericos.pdf>. Acesso em: 09 jun. 2025.