Método de Runge-Kutta de ordem 4: um exemplo usando Python

Prof. Doherty Andrade

www.metodosnumericos.com.br

1 O Método de Runge-Kutta de ordem 4

Vamos estudar numericamente a solução do seguinte problema de valor inicial

$$y' + 2xy = 4x$$

e $y(0) = 1$, no intervalo $[0, 2]$ com passo $h = 0.2$. Faremos isto utilizando o método Runge-Kutta de ordem 4.

Para efeito de comparação vamos usar a solução exata do PVI: a solução exata é $y(x) = 2 - \exp(-x^2)$.

Para entender como se aplica o método, consideremos o PVI genérico

$$\begin{cases} y' = f(x, y) \\ y(x_0) = y_0. \end{cases}$$

Tomamos combinações de $f(x, y)$ em vários pontos do intervalo $x_k \leq x \leq x_{k+1}$ e ajustamos os parâmetros para que nossa aproximação de y_{k+1} coincida com a expansão de Taylor de ordem 4.

Resumindo,

$$y_{k+1} = y_k + \frac{1}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right), \tag{1}$$

onde

$$\begin{aligned} K_1 &= hf(x_k, y_k), \\ K_2 &= hf(x_k + \frac{1}{2}h, y_k + \frac{1}{2}K_1), \\ K_3 &= hf(x_k + \frac{1}{2}h, y_k + \frac{1}{2}K_2), \\ K_4 &= hf(x_k + h, y_k + K_3). \end{aligned}$$

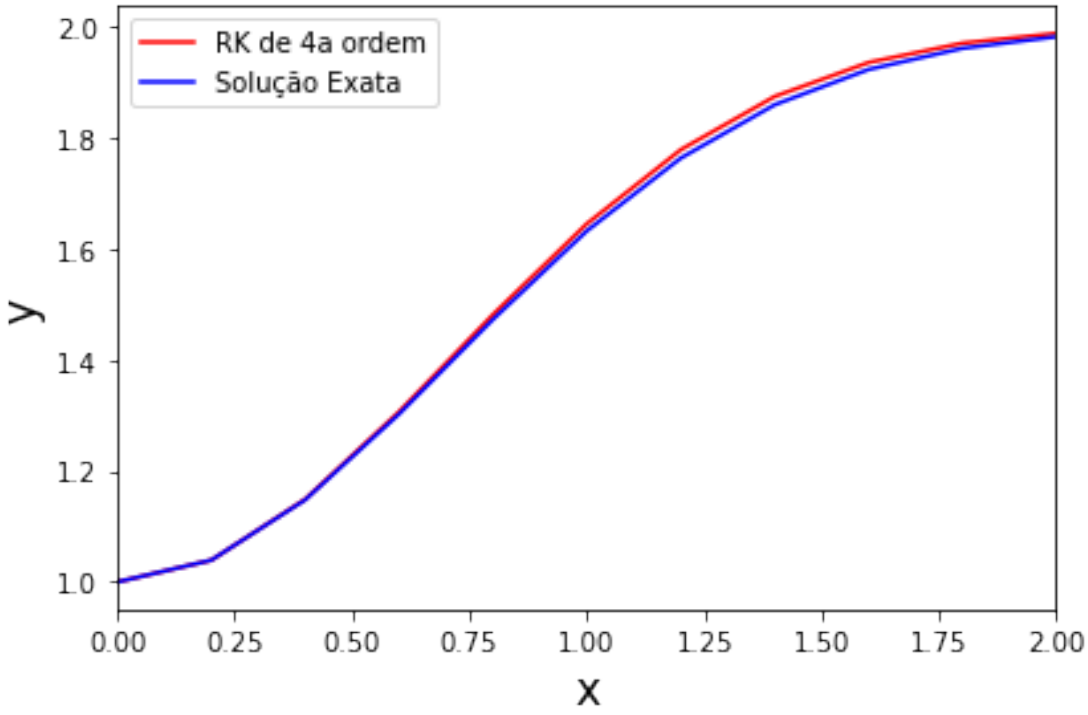
Vamos ao código em Python do script.

2 Um exemplo usando Python

In [50]:
```
#importando as bibliotecas

import matplotlib.pyplot as plt
import numpy as np


def feval(funcName, *args):
    return eval(funcName)(*args)


def RK4Ordem(func, yinit, x_range, h):
    m = len(yinit)
    n = int((x_range[-1] - x_range[0])/h)

    x = x_range[0]
    y = yinit

    xsol = np.empty(0)
    xsol = np.append(xsol, x)

    ysol = np.empty(0)
    ysol = np.append(ysol, y)

    for i in range(n):
        k1 = h*feval(func, x, y)

        k2 = h*feval(func, x+h/2,  y + k1*(h/2))

        k3 = h*feval(func, x+h/2, y + k2*(1/2))

        k4 = h*feval(func, x+h, y + k3)

        for j in range(m):
            y[j] = y[j] + (1/6)*(k1[j] + 2*k2[j] + 2*k3[j] + k4[j])

        x = x + h
        xsol = np.append(xsol, x)

        for r in range(len(y)):
            ysol = np.append(ysol, y[r])

    return [xsol, ysol]
```

```
        def myFunc(x, y):
            dy = np.zeros((len(y)))
            dy[0] = 4*x- 2*x*y
            return dy

        # ----------------------h bem pequeno

        h = 0.2
        x = np.array([0.0, 2.0])
        yinit = np.array([1.0])


        [ts, ys] = RK4Ordem('myFunc', yinit, x, h)

        dt = int((x[-1]-x[0])/h)
        t = [x[0]+i*h for i in range(dt+1)]
        yexact = []
        for i in range(dt+1):
            ye = 2 - 1*np.exp(-t[i]*t[i])
            yexact.append(ye)


        difere = ys - yexact
        print("Diferença máxima =", np.max(abs(difere)))

        plt.plot(ts, ys, 'r')
        plt.plot(t, yexact, 'b')
        plt.xlim(x[0], x[1])
        plt.legend(["RK de 4a ordem", "Solução Exata"], loc=2)
        plt.xlabel('x', fontsize=17)
        plt.ylabel('y', fontsize=17)
        plt.tight_layout()
        plt.show()
```

```
Diferença máxima = 0.015184147077439869
```


RK de 4a ordem
Solução Exata
2.0
1.8
1.6
1.4
1.2
1.0
y
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
x

In [ ]: