# O método de Runge-Kutta-Felberg de ordem 4.5 Exemplo em Python

Prof. Doherty Andrade

www.metodosnumeicos.com.br

## 1 Runge-Kutta-Felberg 4.5

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-Felberg de ordem 4.5. Este é um método em que o passo $h$ não é constante, o método procura adaptá-lo.

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

Consideremos o PVI genérico

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

Resumindo,

$$y_{k+1} = y_k + h\left(\frac{37}{378}K_1 + \frac{250}{621}K_3 + \frac{125}{594}K_4 + \frac{512}{1771}K6\right), \tag{1}$$

onde

$$\begin{aligned}
K_1 &= f(x_k, y_k), \\
K_2 &= f(x_k + \frac{h}{5}, y_k + \frac{h}{5}K_1), \\
K_3 &= f(x_k + \frac{3h}{10}, y_k + \frac{3h}{40}K_1 + \frac{9h}{40}K_2), \\
K_4 &= f(x_k + \frac{3h}{5}, y_k + \frac{3h}{10}K_1 - \frac{9h}{10}K_2 + \frac{6h}{5}K_3), \\
K_5 &= f(x_k + h, y_k - \frac{11h}{54}K_1 + \frac{5h}{2}K_2 - \frac{70h}{27}K_3 + \frac{35h}{27}K_4), \\
K_6 &= f(x_k + \frac{7h}{8}, y_k + \frac{1631h}{55296}K_1 + \frac{175h}{512}K_2 + \frac{575h}{13824}K_3 + \frac{44275h}{110592}K_4 + \frac{253h}{4096}K_5).
\end{aligned}$$

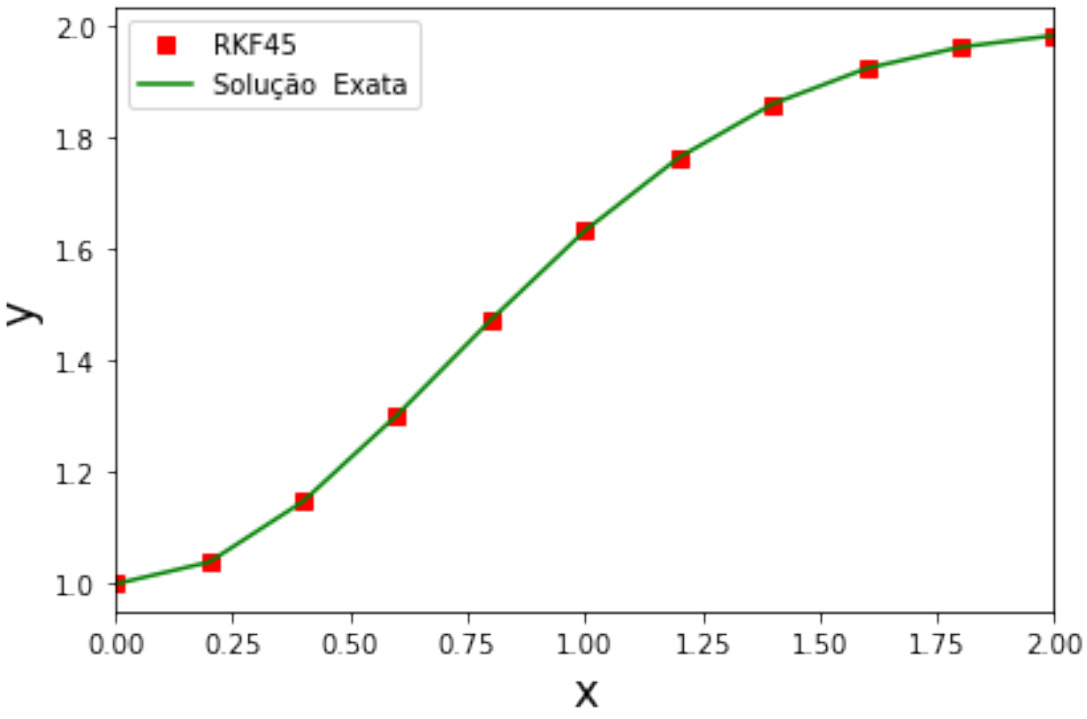
Vejamos o script em Python.

# 2 Exemplo em Python

In [28]:
```
import matplotlib.pyplot as plt
import numpy as np


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


def RKF45(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 = feval(func, x, y)

        k2 = h*feval(func, x+h/5, y + k1*(h/5))

        k3 = feval(func, x+(3*h/10), y + k1*(3*h/40) + k2*(9*h/40))

        k4 = feval(func, x+(3*h/5), y + k1*(3*h/10) - k2*(9*h/10) + k3*(6*h/5))

        k5 = feval(func, x+h, y - k1*(11*h/54) + k2*(5*h/2) - k3*(70*h/27) + k4*(35*h/27))

        k6 = feval(func, x+(7*h/8), y + k1*(1631*h/55296) + k2*(175*h/512) + k3*(575*h/13824) + k4*(44275*h/110592)
                   + k5*(253*h/4096))

        for j in range(m):
            y[j] = y[j] + h*(37*k1[j]/378 + 250*k3[j]/621 + 125*k4[j]/594 + 512*k6[j]/1771)

        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] = RKF45('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- np.exp(- t[i]* t[i])
            yexact.append(ye)


        y_diff = ys - yexact
        print("max diff =", np.max(abs(y_diff)))


        plt.plot(ts, ys, 'rs')
        plt.plot(t, yexact, 'g')
        plt.xlim(x[0], x[1])
        plt.legend(["RKF45", "Solução  Exata"], loc=0)
        plt.xlabel('x', fontsize=17)
        plt.ylabel('y', fontsize=17)
        plt.tight_layout()
        plt.show()
```

```
max diff = 7.917573384341736e-05
```




In [ ]: