O método de Adams-Bashforth-Moulton de quarta ordem
Exemplo com Python

Prof. Doherty Andrade

www.metodosnumeicos.com.br

1 O método

Consideremos, ainda o problema de valor inicial,

$$\begin{cases} y'(x) = f(x, y(x)), \;\; x \in [a, b] \\ y(x_0) = y_0, \end{cases}$$

tomemos os pontos $a = x_0 < x_1 < x_2 < \cdots < x_n = b$ igualmente espaçados e $h = x_{i+1} - x_i$. Integrando,

$$\int_{x_i}^{x_{i+1}} y'(x)dx = \int_{x_i}^{x_{i+1}} f(x, y(x))dx,$$

o que dá

$$y(x_{i+1}) = y(x_i) + \int_{x_i}^{x_{i+1}} f(x, y(x))dx. \tag{1}$$

Os métodos de passo múltiplo são baseados na aproximação da integral acima, substituindo f por um polinômio interpolador em nós (abscissas) equidistantes. O Método Adams-Bashforth de quarta ordem é obtido aproximando f por um polinômio de grau 3 interpolador em $x_i, x_{i-1}, x_{i-2}, x_{i-3}$ e usando a regra de integração

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx \frac{h}{24}\left[55f(x_i) - 59f(x_{i-1}) + 37f(x_{i-2}) - 9f(x_{i-3})\right]$$

onde $h = x_{i+1} - x_i$. Substituindo obtemos:

$$y_{i+1} = y_i + \frac{h}{24}\left[55f(x_i, y_i) - 59f(x_{i-1}, y_{i-1}) + 37f(x_{i-2}, y_{i-2}) - 9f(x_{i-3}, y_{i-3})\right], i \geq 3 \tag{2}$$

que é a fórmula do Método de Adams-Bashforth de quarta ordem.

Observe que para determinar y_4 precisamos conhecer y_0, y_1, y_2, y_3; como y_0 já é dado precisamos determinar os demais. Fazemos isto usando um método numérico de passo simples, aqui usaremos Runge-Kutta de ordem 4.

O método de Adams-Moulton de 4ª ordem e passo 3 é o método implícito dado por

$$y_{i+1} = y_i + \frac{h}{24}\left[9f(x_{i+1}, y_{i+1}) + 19f(x_i, y_i) - 5f(x_{i-1}, y_{i-1}) + f(x_{i-2}, y_{i-2})\right]. \qquad (3)$$

Para aplicar um método implícito diretamente, como o método dado acima, devemos resolver a equação implícita dada pelo método na expressão acima. Isto não é óbvio que possa ser feito, a menos que tenhamos solução.

No caso de solução seguimos os seguintes passos para obter y_{n+1} no método numérico implícito.

(1i) Usando um método explícito determinamos uma aproximação inicial $y_{n+1}^{(0)}$ para y_{n+1},

(2i) Calculamos $f(x_{i+1}, y_{i+1}^{(0)})$,

(3i) Com este valor calculamos $y_{i+1}^{(1)}$ usando o método implícito,

(4i) Voltando em (2i) e calculamos $f(x_{i+1}, y_{i+1}^{(1)})$. Repete-se o procedimento até que

$$\frac{|y_{i+1}^{(k)} - y_{i+1}^{(k-1)})|}{|y_{i+1}^{(k)}|} < \varepsilon.$$

A combinação de um método explícito e um método implícito para resolver problemas de valor inicial é chamado de método previsor-corretor. A fórmula explícita é chamada de previsor e a fórmula implícita é chamada de corretor. Na prática os métodos implícitos de passo múltiplo são usados para melhorar aproximações obtidas pelos métodos explícitos.

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 de Adams-Bashfort-Moulton implícito de ordem 4 e passo 3, como este é um método implícito multi-step iniciamos usando Runge-Kutta de ordem 4.

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

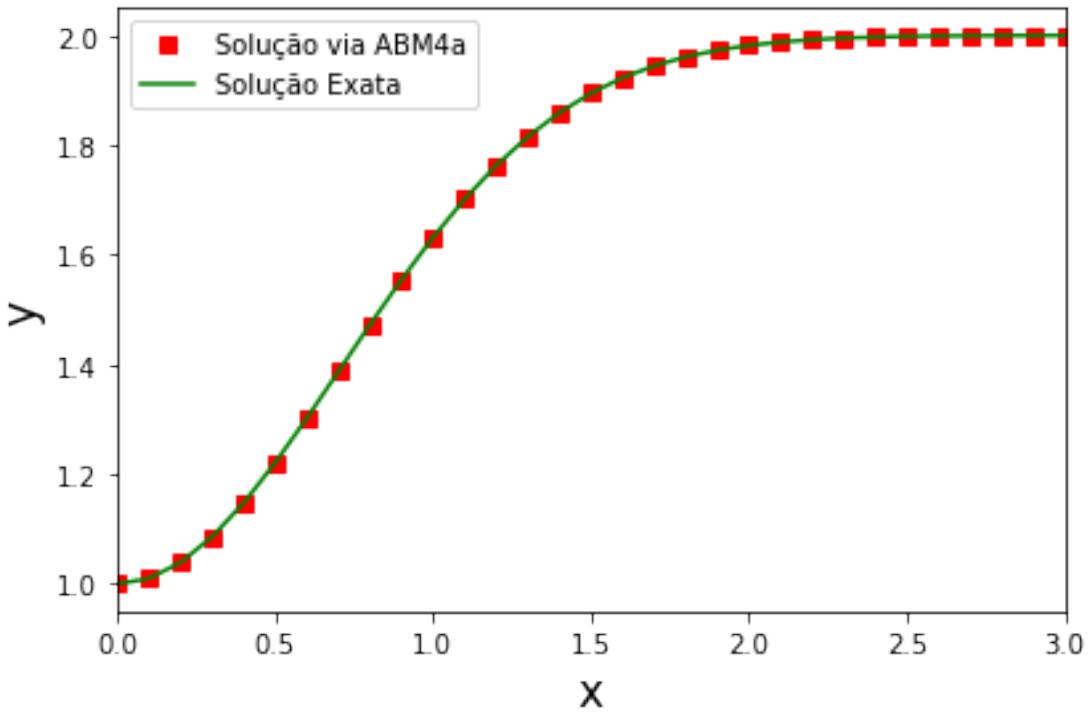
2 Exemplo com Python

```
In [52]: import numpy as np
         import matplotlib.pyplot as plt


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

```
# método Runge-Kutta de 4a ordem para começar
def RungeKutta4aOrdem(func, yinit, xspan, h):
    m = len(yinit)
    n = int((xspan[-1] - xspan[0]) / h)

    x = xspan[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*(1/2))
        k3 = h*feval(func, x+h/2, y + k2*(1/2))
        k4 = h*feval(func, x+h, y + k3*h)
        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]

# metodo de Adams-Bashfort-Moulton de 4a ordem
def ABM4aOrdem(func, yinit, xspan, h):

    m = len(yinit)

    dx = int((xspan[-1] - xspan[0]) / h)

    xrk = [xspan[0] + k * h for k in range(dx + 1)]

    [xx, yy] = RungeKutta4aOrdem(func, yinit, (xrk[0], xrk[3]), h)

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

    y = yy
    yn = np.array([yy[0]])
    ysol = np.empty(0)
    ysol = np.append(ysol, y)

    for i in range(3, dx):
        x00 = x[i]; x11 = x[i-1]; x22 = x[i-2]; x33 = x[i-3]; xpp = x[i]+h

        y00 = np.array([y[i]])
        y11 = np.array([y[i - 1]])
        y22 = np.array([y[i - 2]])
        y33 = np.array([y[i - 3]])

        y0prime = feval(func, x00, y00)
        y1prime = feval(func, x11, y11)
        y2prime = feval(func, x22, y22)
        y3prime = feval(func, x33, y33)
#preditor
        ypredictor = y00 + (h/24)*(55*feval(func, x00, y00) - 59*feval(func, x11, y11) + \
                                   37*feval(func, x22, y22) - 9*feval(func, x33, y33))
        ypp = feval(func, xpp, ypredictor)
#corretor
        for j in range(m):
```

```
                yn[j] = y00[j] + (h/24)*(9*ypp[j] + 19*y0prime[j] - 5*y1prime[j] + y2prime[j])

            xs = x[i] + h
            xsol = np.append(xsol, xs)

            x = xsol

            for r in range(len(yn)):
                ysol = np.append(ysol, yn)

            y = ysol

        return [xsol, ysol]

    # a f do pvi estudado
    def myFunc(x, y):
        dy = np.zeros((len(y)))
        dy[0] = 4*x - 2*x*y
        return dy

    #Escolhendo h
    h = 0.1
    xspan = np.array([0.0, 3.0])
    yinit = np.array([1.0])

    #aplicando o metodo ABM4
    [ts, ys] = ABM4aOrdem('myFunc', yinit, xspan, h)


    dt = int((xspan[-1]-xspan[0])/h)
    t = [xspan[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)

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

    #para ver a solucao numerica
    print('Solução numérica é:','ts:', ts)
    print('Solução numérica é:','ys:', ys)
    #plotando as soluções
    plt.plot(ts, ys, 'rs')
    plt.plot(t, yexact, 'g')
    plt.xlim(xspan[0], xspan[1])
    plt.legend(["Solução via ABM4a", "Solução Exata"], loc=2)
    plt.xlabel('x', fontsize=17)
    plt.ylabel('y', fontsize=17)
    plt.tight_layout()
    plt.show()
```

```
Diferença Máxima = 0.000616234279086969
Solução numérica é: ts: [0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5 1.6 1.7
 1.8 1.9 2.  2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3. ]
Solução numérica é: ys: [1.         1.00998002 1.03941509 1.08668505 1.14843763 1.22174242
 1.30282391 1.38782612 1.47310876 1.55548927 1.63241297 1.70204092
 1.76325887 1.81562007 1.85924026 1.8946657  1.92273398 1.94444361
 1.96084353 1.97294851 1.98168163 1.98784164 1.99209081 1.99495775
 1.99685003 1.99807196 1.99884396 1.99932117 1.99960975 1.99978045
 1.99987919]
```


Solução via ABM4a
Solução Exata
2.0
1.8
1.6
1.4
1.2
1.0
y
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x

In [ ]: