Nathan Martins
Métodos Numéricos Computacionais
09 November 2017

Relatório de Implementações

MÉTODOS IMPLEMENTADOS

- Euler Simples
- Euler Aprimorado
- Euler Inverso
- Runge-Kutta
- Adams-Bashforth (Ordens 1 a 6)
- Adams-Moulton (Ordens 1 a 6)

ALGORITMO BASE DE TODOS OS MÉTODOS

Todas as implementações seguem os seguintes passos:

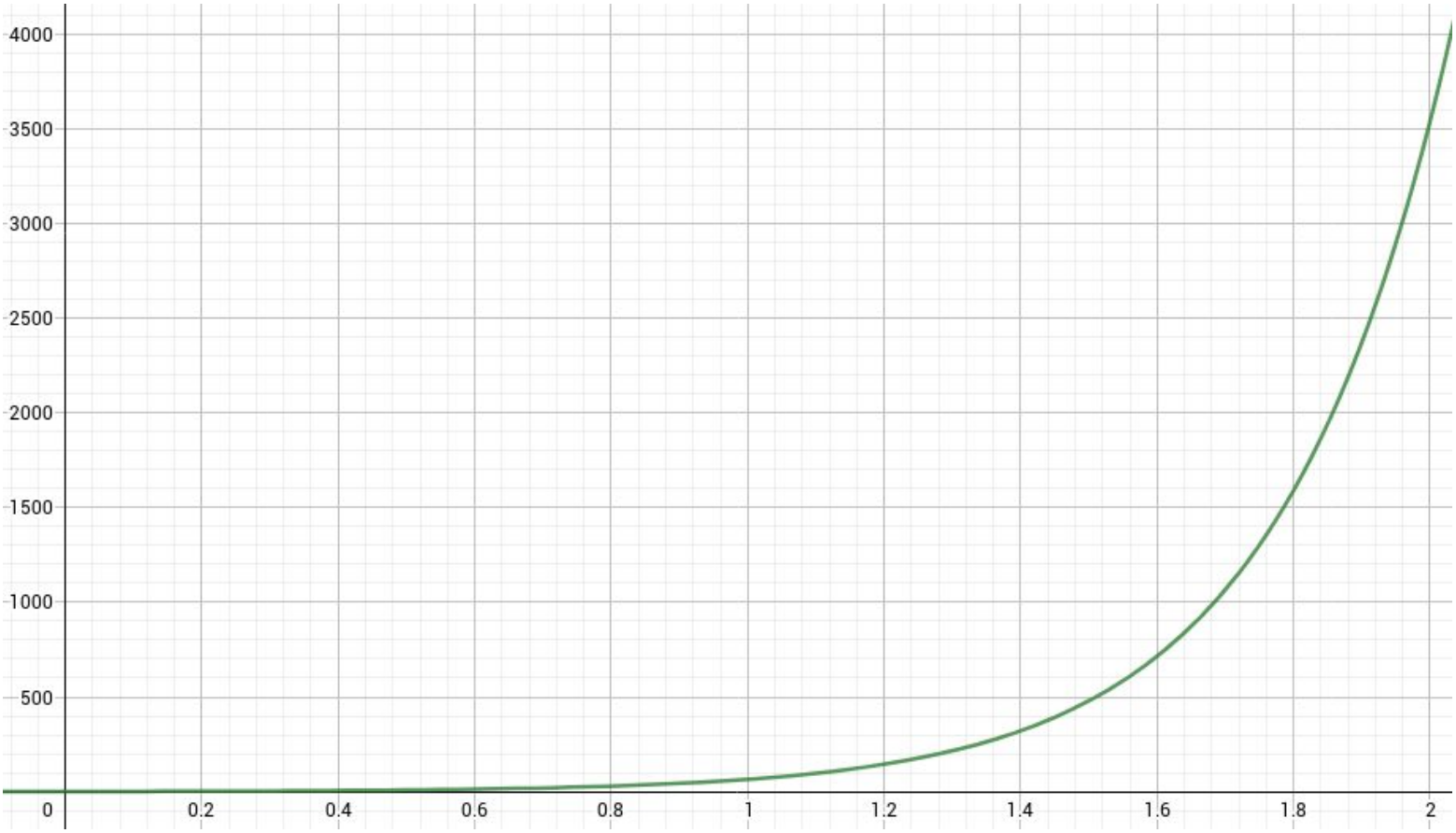
1. Recebem a função $f(t, y)$
2. Recebem os pontos (t_i, y_i)
3. Recebem o tamanho do passo $h = t_{i+1} - t_i$
4. Recebem o número de passos n
5. Faz n repetições de um algoritmo particular
6. Abrem um plot da função e imprime os pontos no terminal

ALGORITMO PARTICULAR DE CADA MÉTODO

Para os exemplos a seguir a seguinte configuração será utilizada:

$y(t) = 1 - t + 4y$
$y(0) = 1$
$h = .05$
$n = 40$

Aqui está um plot da solução exata para comparação:



Fontes: WolframAlpha e Geogebra

Euler Simples

Este método tenta aproximar y(t) com a seguinte fórmula:

$$y_{n+1} = y_n + f(t_n, y_n)(t_{n+1} - t_n)$$

Onde

$$f(t, y) = \frac{dy(t)}{dt}$$

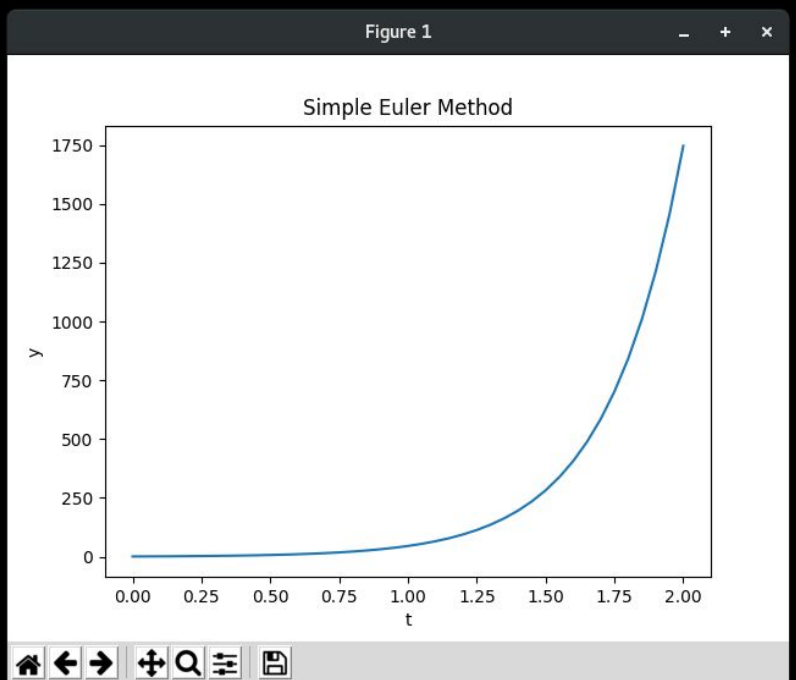
O método simplesmente calcula o próximo ponto utilizando o ponto atual mais um certo passo vezes a inclinação da função y(t) naquele ponto, por isso este método é conhecido como método da reta tangente, ele calcula o próximo ponto com a reta tangente ao ponto atual. Este método tende a "subestimar" o valor exato da função, resultando sempre em valores menores que o real.

O algoritmo particular para este método é a repetição dos seguintes passos:

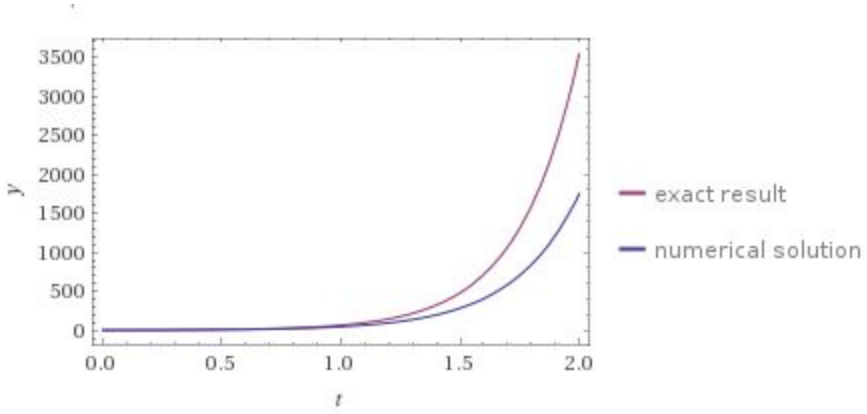
1. $k1 = f(t, y)$
2. $y = y + h * k1$
3. $t = t + h$

Exemplo:

```
Running Simple Euler Method
y(0.0500) = 1.2500000
y(0.1000) = 1.5475000
y(0.1500) = 1.9020000
y(0.2000) = 2.3249000
y(0.2500) = 2.8298800
y(0.3000) = 3.4333560
y(0.3500) = 4.1550272
y(0.4000) = 5.0185326
y(0.4500) = 6.0522392
y(0.5000) = 7.2901870
y(0.5500) = 8.7732244
y(0.6000) = 10.5503693
y(0.6500) = 12.6804431
y(0.7000) = 15.2340318
y(0.7500) = 18.2958381
y(0.8000) = 21.9675057
y(0.8500) = 26.3710069
y(0.9000) = 31.6527083
y(0.9500) = 37.9882499
y(1.0000) = 45.5883999
y(1.0500) = 54.7060799
y(1.1000) = 65.6447959
y(1.1500) = 78.7687550
y(1.2000) = 94.5150061
y(1.2500) = 113.4080073
y(1.3000) = 136.0771087
y(1.3500) = 163.2775305
y(1.4000) = 195.9155366
y(1.4500) = 235.0786439
y(1.5000) = 282.0718726
y(1.5500) = 338.4612472
y(1.6000) = 406.1259966
y(1.6500) = 487.3211959
y(1.7000) = 584.7529351
y(1.7500) = 701.6685221
y(1.8000) = 841.9647265
y(1.8500) = 1010.3176719
y(1.9000) = 1212.3387062
y(1.9500) = 1454.7614475
y(2.0000) = 1745.6662370
```



A título de comparação: Fonte (Wolfram Alpha)



Euler Inverso

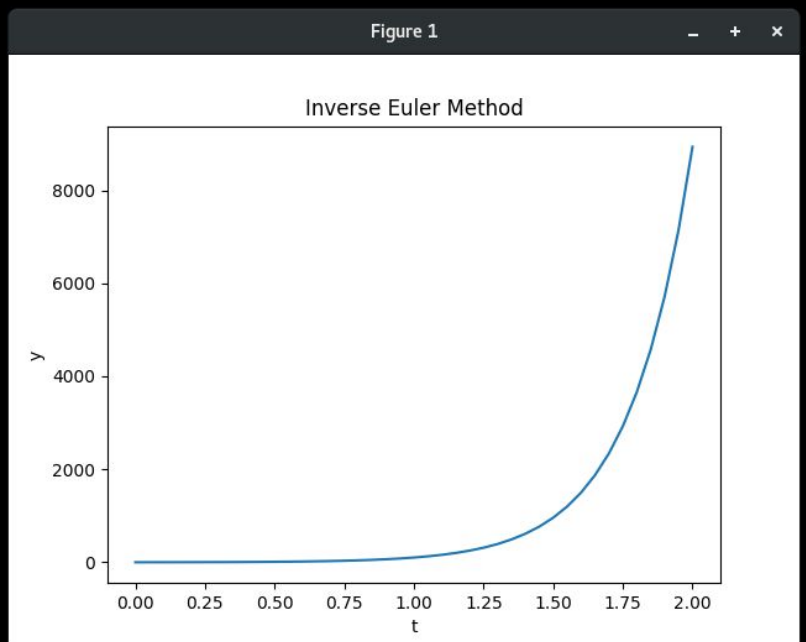
Este método é bem parecido com o Euler Simples porém ele é um método implícito. A parcela y_{n+1} aparece dos dois lados da equação:

$$y_{n+1} = y_n + f(t_{n+1}, y_{n+1})(t_{n+1} - t_n)$$

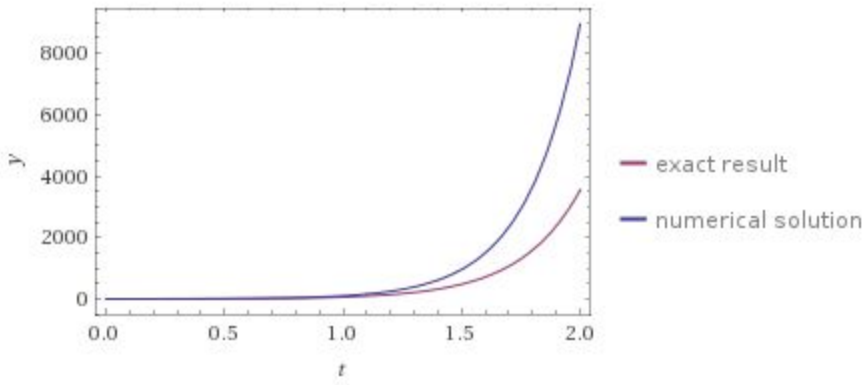
Ao invés de utilizar a inclinação da reta no ponto n, é utilizada a inclinação no ponto n+1. Isso faz com que a cada iteração uma equação algébrica precise ser resolvida, o que deixa o método mais lento. Ainda, a aproximação com esse método tende a crescer muito rápido, como pode ser notado com a segunda figura, fica muito distante do valor real muito rapidamente.

1. $y_{n+1} = y_n + h * f(t_{n+1}, y_{n+1})$
2. $t = t + h$

```
Running Inverse Euler Method
y(0.0500) = 1.3093750
y(0.1000) = 1.6929688
y(0.1500) = 2.1693359
y(0.2000) = 2.7616699
y(0.2500) = 3.4989624
y(0.3000) = 4.4174530
y(0.3500) = 5.5624413
y(0.4000) = 6.9905516
y(0.4500) = 8.7725645
y(0.5000) = 10.9969556
y(0.5500) = 13.7743195
y(0.6000) = 17.2428993
y(0.6500) = 21.5754992
y(0.7000) = 26.9881240
y(0.7500) = 33.7507799
y(0.8000) = 42.2009749
y(0.8500) = 52.7605937
y(0.9000) = 65.9569921
y(0.9500) = 82.4493651
y(1.0000) = 103.0617064
y(1.0500) = 128.8240080
y(1.1000) = 161.0237600
y(1.1500) = 201.2703250
y(1.2000) = 251.5754062
y(1.2500) = 314.4536328
y(1.3000) = 393.0482910
y(1.3500) = 491.2884887
y(1.4000) = 614.0856109
y(1.4500) = 767.5788886
y(1.5000) = 959.4423607
y(1.5500) = 1199.2685759
y(1.6000) = 1499.0482199
y(1.6500) = 1873.7696499
y(1.7000) = 2342.1683124
y(1.7500) = 2927.6635155
y(1.8000) = 3659.5293943
y(1.8500) = 4574.3586179
y(1.9000) = 5717.8920224
y(1.9500) = 7147.3056530
y(2.0000) = 8934.0695662
```



Comparando: [Fonte]

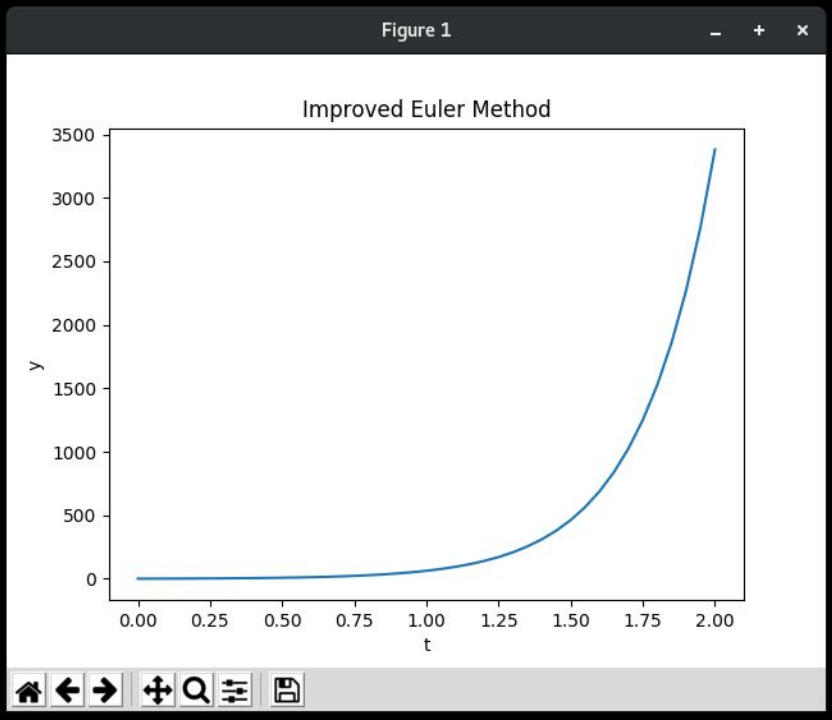


Euler Aprimorado (Heun)

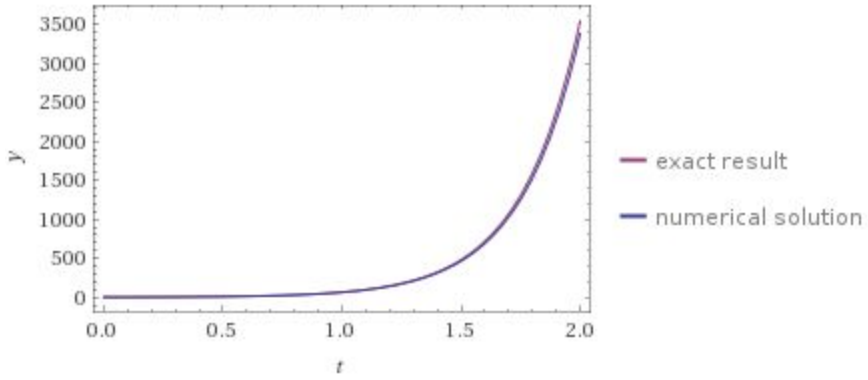
Este método é basicamente uma junção dos dois métodos anteriores. Uma vez que o Euler Simples “subestima” o valor exato da função e o Euler Aprimorado “superestima” o valor exato, este método faz a média entre os valores das inclinações das retas tangentes calculadas em n e n+1 para conseguir uma melhor aproximação da reta tangente. Isto pode ser visto na terceira figura.

1. $y_{n+1} = y_n + \frac{f_n + f(t_n + h,\ y_n + hf_n)}{2} h$
2. $t = t + h$

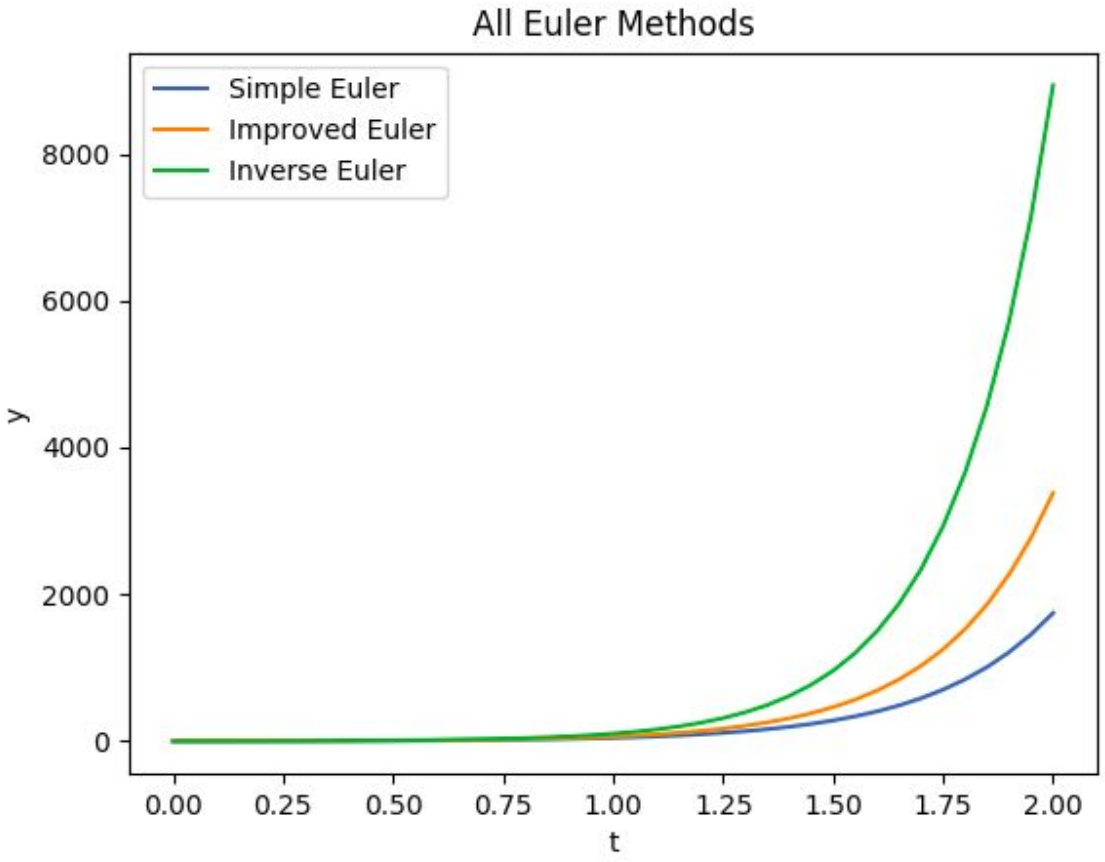
```
Running Improved Euler Method
y(0.0500) = 1.2737500
y(0.1000) = 1.6049750
y(0.1500) = 2.0063195
y(0.2000) = 2.4932098
y(0.2500) = 3.0844659
y(0.3000) = 3.8030485
y(0.3500) = 4.6769691
y(0.4000) = 5.7404023
y(0.4500) = 7.0350408
y(0.5000) = 8.6117498
y(0.5500) = 10.5325848
y(0.6000) = 12.8732534
y(0.6500) = 15.7261192
y(0.7000) = 19.2038654
y(0.7500) = 23.4439658
y(0.8000) = 28.6141382
y(0.8500) = 34.9189986
y(0.9000) = 42.6081783
y(0.9500) = 51.9862276
y(1.0000) = 63.4246976
y(1.0500) = 77.3768811
y(1.1000) = 94.3957950
y(1.1500) = 115.1561199
y(1.2000) = 140.4809662
y(1.2500) = 171.3745288
y(1.3000) = 209.0619251
y(1.3500) = 255.0377987
y(1.4000) = 311.1256144
y(1.4500) = 379.5499995
y(1.5000) = 463.0249994
y(1.5500) = 564.8617493
y(1.6000) = 689.0998341
y(1.6500) = 840.6675476
y(1.7000) = 1025.5774081
y(1.7500) = 1251.1646879
y(1.8000) = 1526.3784192
y(1.8500) = 1862.1364215
y(1.9000) = 2271.7584342
y(1.9500) = 2771.4945397
y(2.0000) = 3381.1698384
```



Comparação: [Fonte]



Comparação entre os Eulers:



Runge-Kutta

Este método faz uma média ponderada de várias inclinações da reta tangente dentro do intervalo de tempo $h = t_{i+1} - t_i$.

Como pode ser visto abaixo o k1 é simplesmente o k do método de Euler Simples. O k2 utiliza o k2 para estimar a inclinação da reta no ponto médio do intervalo de tempo dado pelo passo h ($t_n + h/2$), o k3 utiliza o k2 para fazer a mesma coisa que o k2 faz com o k1, estima a inclinação da reta no ponto médio do intervalo de tempo. O k4 utiliza o k3 para estimar a inclinação da reta no fim do intervalo de tempo dado pelo passo ($t_n + h/2 = t_{n+1}$).

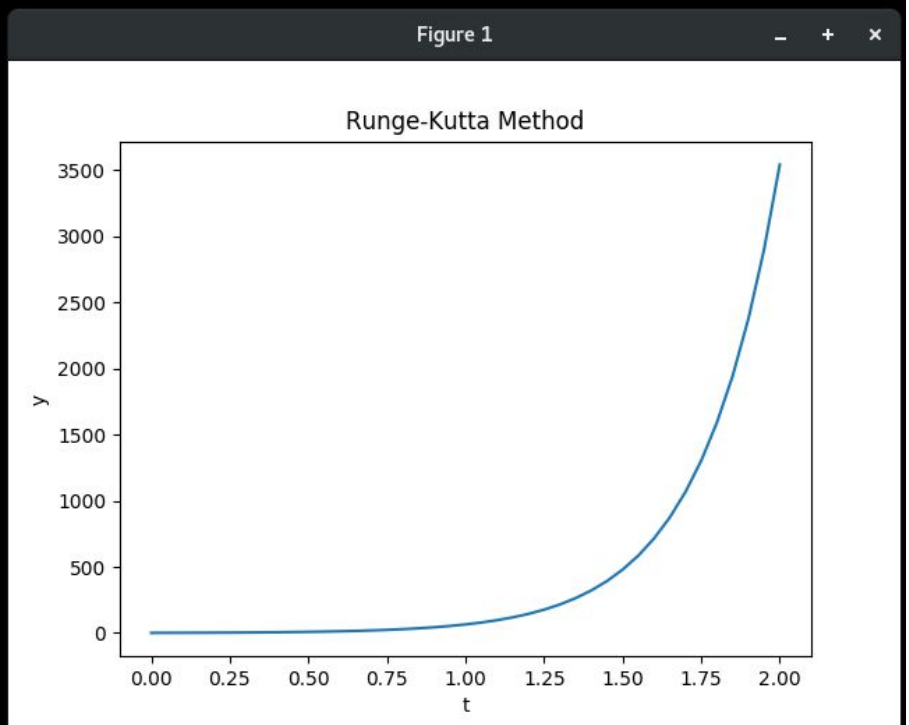
Após isso é feita a média ponderada dessas inclinações dando mais peso as inclinações centrais (k2 e k3), pois são as que tem mais chances de chegar perto do valor exato (como visto anteriormente com o método de Euler Aprimorado), e só então o novo ponto é calculado.

Segue o algoritmo:

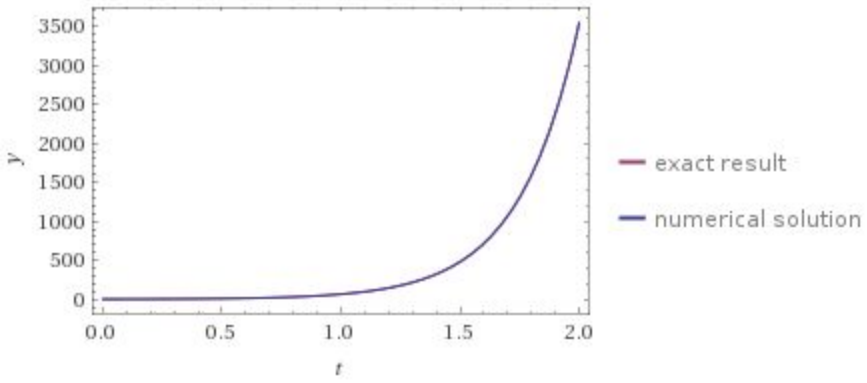
$$k_1 = f(t_n, y_n),$$
$$k_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right),$$
$$k_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right),$$
$$k_4 = f\left(t_n + h, y_n + hk_3\right).$$
$$y_{n+1} = y_n + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right),$$
$$t_{n+1} = t_n + h$$

Exemplo:

```
Running Runge-Kutta
y(0.0500) = 1.2754125
y(0.1000) = 1.6090338
y(0.1500) = 2.0137514
y(0.2000) = 2.5053060
y(0.2500) = 3.1029232
y(0.3000) = 3.8300854
y(0.3500) = 4.7154738
y(0.4000) = 5.7941197
y(0.4500) = 7.1088104
y(0.5000) = 8.7118060
y(0.5500) = 10.6669373
y(0.6000) = 13.0521672
y(0.6500) = 15.9627196
y(0.7000) = 19.5149007
y(0.7500) = 23.8507672
y(0.8000) = 29.1438270
y(0.8500) = 35.6060029
y(0.9000) = 43.4961369
y(0.9500) = 53.1303791
y(1.0000) = 64.8948750
y(1.0500) = 79.2612628
y(1.1000) = 96.8056014
y(1.1500) = 118.2314891
y(1.2000) = 144.3983008
y(1.2500) = 176.3556771
y(1.3000) = 215.3856490
y(1.3500) = 263.0540892
y(1.4000) = 321.2735545
y(1.4500) = 392.3800420
y(1.5000) = 479.2267383
y(1.5500) = 585.2985256
y(1.6000) = 714.8518392
y(1.6500) = 873.0854889
y(1.7000) = 1066.3493012
y(1.7500) = 1302.3989539
y(1.8000) = 1590.7072323
y(1.8500) = 1942.8441961
y(1.9000) = 2372.9415161
y(1.9500) = 2898.2596152
y(2.0000) = 3539.8803741
```



Comparação: Fonte



Adams-Bashforth

Os métodos anteriores tentam aproximar o valor do próximo ponto utilizando uma reta, métodos de passo múltiplo como o Adams-Bashforth tentam aproximar o valor do próximo ponto utilizando um polinômio.

A intuição é simples: todos os métodos descritos anteriormente estão tentando aproximar uma integral entre dois pontos, pois sabemos que a área abaixo de uma curva é a integral dessa curva, ou seja, a área abaixo da curva de y'(t) é numericamente igual a y(t). Colocando polinômios de interpolação de grau superior, esses métodos conseguem uma precisão bem melhor que os anteriores.

O diferencial aqui é que cada iteração do método precisa de uma quantidade de pontos que é igual ao grau do método. Ou seja, um método de grau quatro utiliza quatro pontos e esse conjunto de pontos é sempre atualizado, por exemplo: Se os pontos utilizados em uma iteração foram p1, p2, p3 e p4 então é obtido um novo ponto p5, para calcular o ponto p6 o ponto p1 é descartado e os pontos p2, p3, p4 e p5 são utilizados. Ou seja, mesmo que mais pontos sejam utilizados apenas um ponto é calculado a cada iteração.

O grau do método depende do grau do polinômio, um polinômio de interpolação de grau seis acarreta num método de grau seis, ou seja, seis pontos iniciais são necessários.

Segue as fórmulas para os graus de um a 6:

$$y_{n+1} = y_n + hf(t_n, y_n), \qquad \text{(This is the Euler method)}$$

$$y_{n+2} = y_{n+1} + h\left(\frac{3}{2}f(t_{n+1}, y_{n+1}) - \frac{1}{2}f(t_n, y_n)\right),$$

$$y_{n+3} = y_{n+2} + h\left(\frac{23}{12}f(t_{n+2}, y_{n+2}) - \frac{4}{3}f(t_{n+1}, y_{n+1}) + \frac{5}{12}f(t_n, y_n)\right),$$

$$y_{n+4} = y_{n+3} + h\left(\frac{55}{24}f(t_{n+3}, y_{n+3}) - \frac{59}{24}f(t_{n+2}, y_{n+2}) + \frac{37}{24}f(t_{n+1}, y_{n+1}) - \frac{3}{8}f(t_n, y_n)\right),$$

$$y_{n+5} = y_{n+4} + h\left(\frac{1901}{720}f(t_{n+4}, y_{n+4}) - \frac{1387}{360}f(t_{n+3}, y_{n+3}) + \frac{109}{30}f(t_{n+2}, y_{n+2}) - \frac{637}{360}f(t_{n+1}, y_{n+1}) + \frac{251}{720}f(t_n, y_n)\right).$$

$$y_{n+6} = y_{n+5} + \frac{h}{1440}(4277f_{n+5} - 7923f_{n+4} + 9982f_{n+3} - 7298f_{n+2} + 2877f_{n+1} - 475f_n)$$

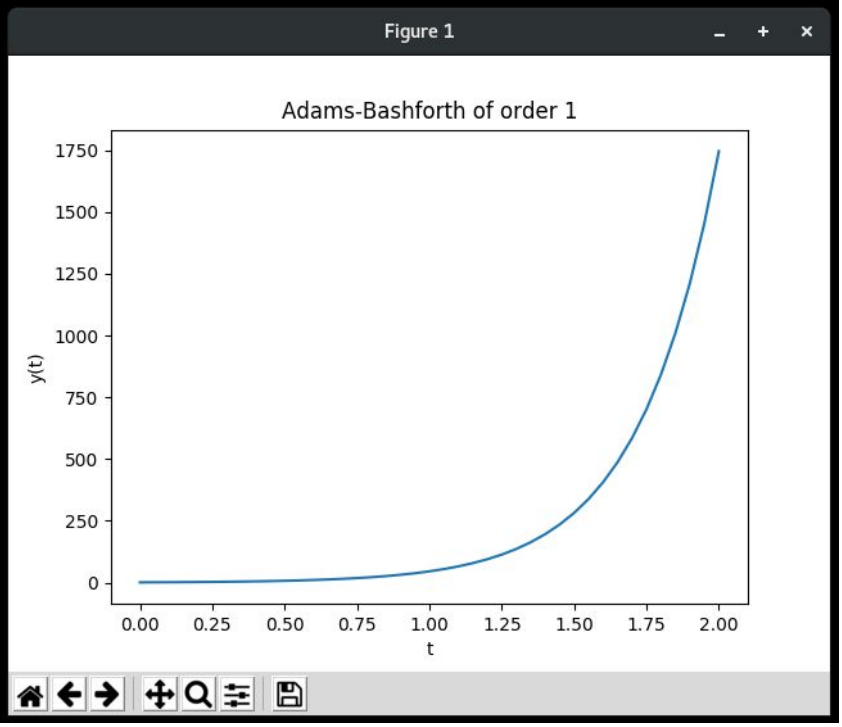
Onde:

$$f_{n+i} = f(t_{n+i}, y_{n+i})$$

Segue as implementações para cada ordem do método:

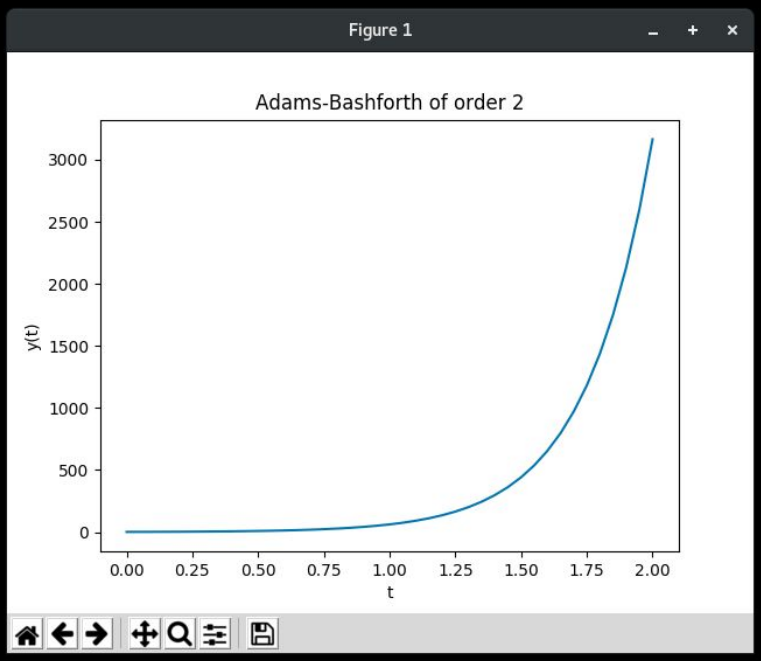
Ordem 1

```
Running Adams-Bashforth of order 1 for 40 iterations
y(0.0000) = 1.0000000
y(0.0500) = 1.2500000
y(0.1000) = 1.5475000
y(0.1500) = 1.9020000
y(0.2000) = 2.3249000
y(0.2500) = 2.8298800
y(0.3000) = 3.4333560
y(0.3500) = 4.1550272
y(0.4000) = 5.0185326
y(0.4500) = 6.0522392
y(0.5000) = 7.2901870
y(0.5500) = 8.7732244
y(0.6000) = 10.5503693
y(0.6500) = 12.6804431
y(0.7000) = 15.2340318
y(0.7500) = 18.2958381
y(0.8000) = 21.9675057
y(0.8500) = 26.3710069
y(0.9000) = 31.6527083
y(0.9500) = 37.9882499
y(1.0000) = 45.5883999
y(1.0500) = 54.7060799
y(1.1000) = 65.6447959
y(1.1500) = 78.7687550
y(1.2000) = 94.5150061
y(1.2500) = 113.4080073
y(1.3000) = 136.0771087
y(1.3500) = 163.2775305
y(1.4000) = 195.9155366
y(1.4500) = 235.0786439
y(1.5000) = 282.0718726
y(1.5500) = 338.4612472
y(1.6000) = 406.1259966
y(1.6500) = 487.3211959
y(1.7000) = 584.7529351
y(1.7500) = 701.6685221
y(1.8000) = 841.9647265
y(1.8500) = 1010.3176719
y(1.9000) = 1212.3387062
y(1.9500) = 1454.7614475
y(2.0000) = 1745.6662370
```



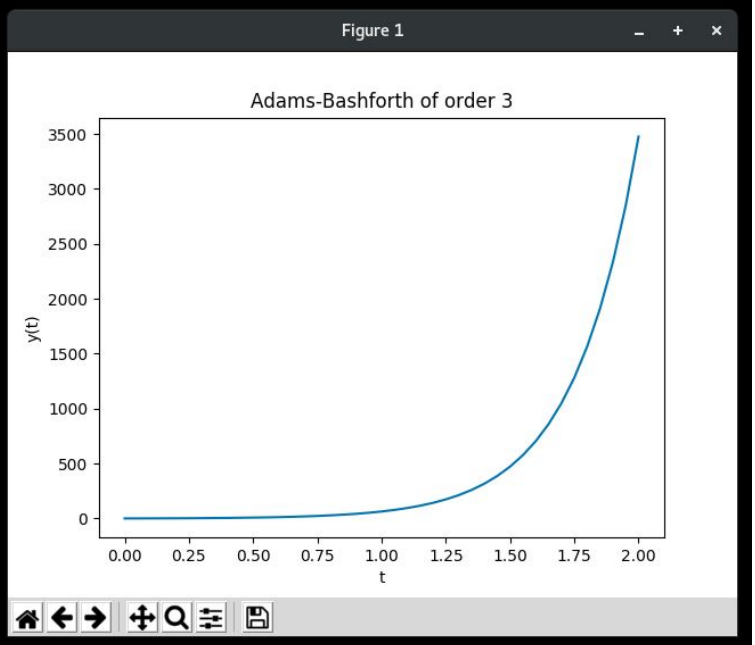
Ordem 2

```
Running Adams-Bashforth of order 2 for 39 iterations
y(0.0000) = 1.0000000
y(0.0500) = 1.2754125
y(0.1000) = 1.6042862
y(0.1500) = 2.0017809
y(0.2000) = 2.4831365
y(0.2500) = 3.0666494
y(0.3000) = 3.7745805
y(0.3500) = 4.6340398
y(0.4000) = 5.6780436
y(0.4500) = 6.9468028
y(0.5000) = 8.4892892
y(0.5500) = 10.3651457
y(0.6000) = 12.6470105
y(0.6500) = 15.4233491
y(0.7000) = 18.8019028
y(0.7500) = 22.9138887
y(0.8000) = 27.9191150
y(0.8500) = 34.0122106
y(0.9000) = 41.4302123
y(0.9500) = 50.4618050
y(1.0000) = 61.4585752
y(1.0500) = 74.8487173
y(1.1000) = 91.1537249
y(1.1500) = 111.0087207
y(1.2000) = 135.1872144
y(1.2500) = 164.6312567
y(1.3000) = 200.4881622
y(1.3500) = 244.1552352
y(1.4000) = 297.3342396
y(1.4500) = 362.0977379
y(1.5000) = 440.9698853
y(1.5500) = 537.0248271
y(1.6000) = 654.0065367
y(1.6500) = 796.4747650
y(1.7000) = 969.9827909
y(1.7500) = 1181.2939016
y(1.8000) = 1438.6450430
y(1.8500) = 1752.0679158
y(1.9000) = 2133.7800362
y(1.9500) = 2598.6610055
y(2.0000) = 3164.8325535
```



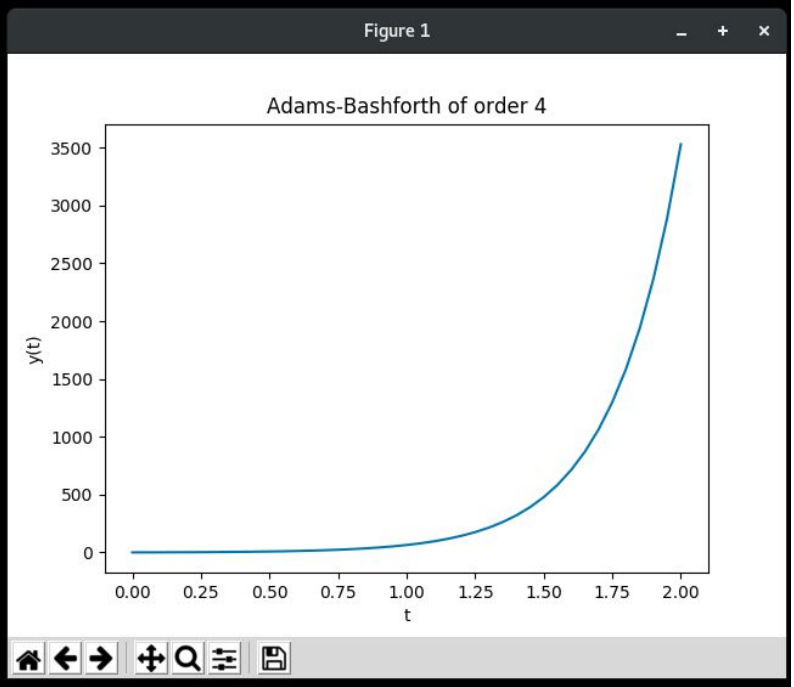
Ordem 3

```
Running Adams-Bashforth of order 3 for 38 iterations
y(0.0000) = 1.0000000
y(0.0500) = 1.2754125
y(0.1000) = 1.6090338
y(0.1500) = 2.0128035
y(0.2000) = 2.5028368
y(0.2500) = 3.0983461
y(0.3000) = 3.8226060
y(0.3500) = 4.7040324
y(0.4000) = 5.7773287
y(0.4500) = 7.0848632
y(0.5000) = 8.6783592
y(0.5500) = 10.6209607
y(0.6000) = 12.9897551
y(0.6500) = 15.8788517
y(0.7000) = 19.4031402
y(0.7500) = 23.7028797
y(0.8000) = 28.9493006
y(0.8500) = 35.3514429
y(0.9000) = 43.1645058
y(0.9500) = 52.7000400
y(1.0000) = 64.3383907
y(1.0500) = 78.5438886
y(1.1000) = 95.8833950
y(1.1500) = 117.0489420
y(1.2000) = 142.8853719
y(1.2500) = 174.4240794
y(1.3000) = 212.9242059
y(1.3500) = 259.9229280
y(1.4000) = 317.2968521
y(1.4500) = 387.3369651
y(1.5000) = 472.8401351
y(1.5500) = 577.2208172
y(1.6000) = 704.6474249
y(1.6500) = 860.2088145
y(1.7000) = 1050.1175315
y(1.7500) = 1281.9579367
y(1.8000) = 1564.9891220
y(1.8500) = 1910.5147132
y(1.9000) = 2332.3343322
y(1.9500) = 2847.2947461
y(2.0000) = 3475.9627197
```



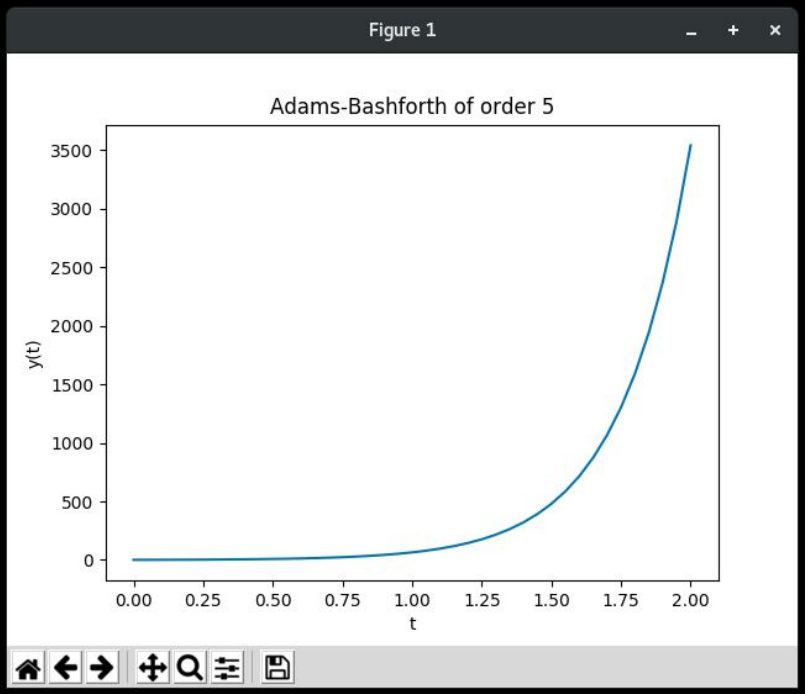
Ordem 4

```
Running Adams-Bashforth of order 4 for 37 iterations
y(0.0000) = 1.0000000
y(0.0500) = 1.2754125
y(0.1000) = 1.6090338
y(0.1500) = 2.0137514
y(0.2000) = 2.5051147
y(0.2500) = 3.1024107
y(0.3000) = 3.8291466
y(0.3500) = 4.7139493
y(0.4000) = 5.7917886
y(0.4500) = 7.1053894
y(0.5000) = 8.7069285
y(0.5500) = 10.6601263
y(0.6000) = 13.0428052
y(0.6500) = 15.9500109
y(0.7000) = 19.4978227
y(0.7500) = 23.8280081
y(0.8000) = 29.1137087
y(0.8500) = 35.5663824
y(0.9000) = 43.4442833
y(0.9500) = 53.0628181
y(1.0000) = 64.8071935
y(1.0500) = 79.1478636
y(1.1000) = 96.6593953
y(1.1500) = 118.0435085
y(1.2000) = 144.1572156
y(1.2500) = 176.0471882
y(1.3000) = 214.9917290
y(1.3500) = 262.5520322
y(1.4000) = 320.6347887
y(1.4500) = 391.5686451
y(1.5000) = 478.1975832
y(1.5500) = 583.9949658
y(1.6000) = 713.2028197
y(1.6500) = 871.0019437
y(1.7000) = 1063.7196606
y(1.7500) = 1299.0835464
y(1.8000) = 1586.5313098
y(1.8500) = 1937.5892494
y(1.9000) = 2366.3344637
y(1.9500) = 2889.9593498
y(2.0000) = 3529.4610274
```



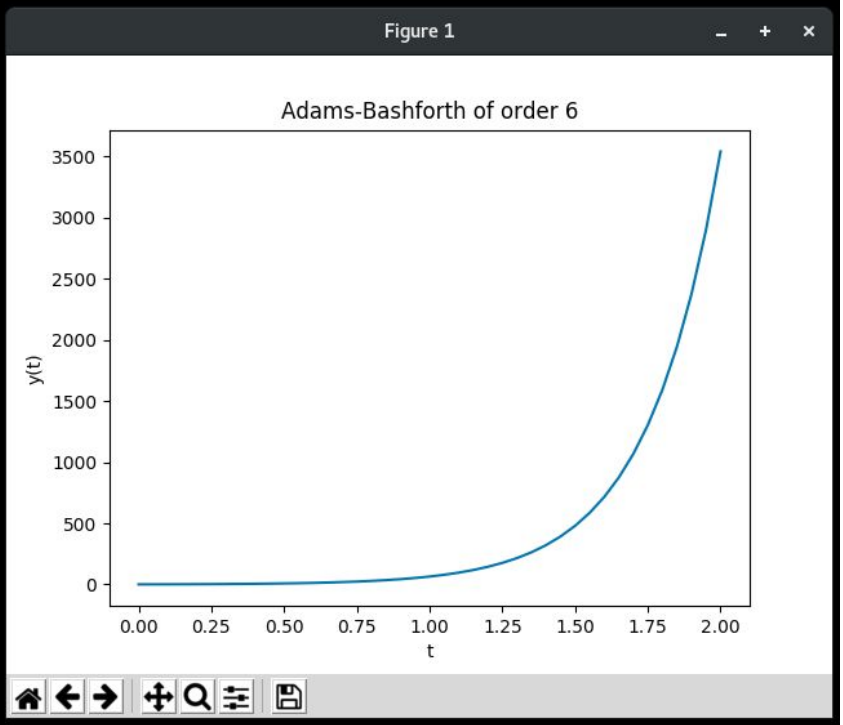
Ordem 5

```
Running Adams-Bashforth of order 5 for 36 iterations
y(0.0000) = 1.0000000
y(0.0500) = 1.2754125
y(0.1000) = 1.6090338
y(0.1500) = 2.0137514
y(0.2000) = 2.5053060
y(0.2500) = 3.1028885
y(0.3000) = 3.8299901
y(0.3500) = 4.7153031
y(0.4000) = 5.7938439
y(0.4500) = 7.1083862
y(0.5000) = 8.7111834
y(0.5500) = 10.6660509
y(0.6000) = 13.0509294
y(0.6500) = 15.9610177
y(0.7000) = 19.5125908
y(0.7500) = 23.8476634
y(0.8000) = 29.1396908
y(0.8500) = 35.6005291
y(0.9000) = 43.4889363
y(0.9500) = 53.1209553
y(1.0000) = 64.8825965
y(1.0500) = 79.2453275
y(1.1000) = 96.7849919
y(1.1500) = 118.2049168
y(1.2000) = 144.3641356
y(1.2500) = 176.3118595
y(1.3000) = 215.3295797
y(1.3500) = 262.9824909
y(1.4000) = 321.1822995
y(1.4500) = 392.2639359
y(1.5000) = 479.0792503
y(1.5500) = 585.1114512
y(1.6000) = 714.6148791
y(1.6500) = 872.7857238
y(1.7000) = 1065.9705365
y(1.7500) = 1301.9209029
y(1.8000) = 1590.1044972
y(1.8500) = 1942.0850002
y(1.9000) = 2371.9861239
y(1.9500) = 2897.0583656
y(2.0000) = 3538.3712331
```

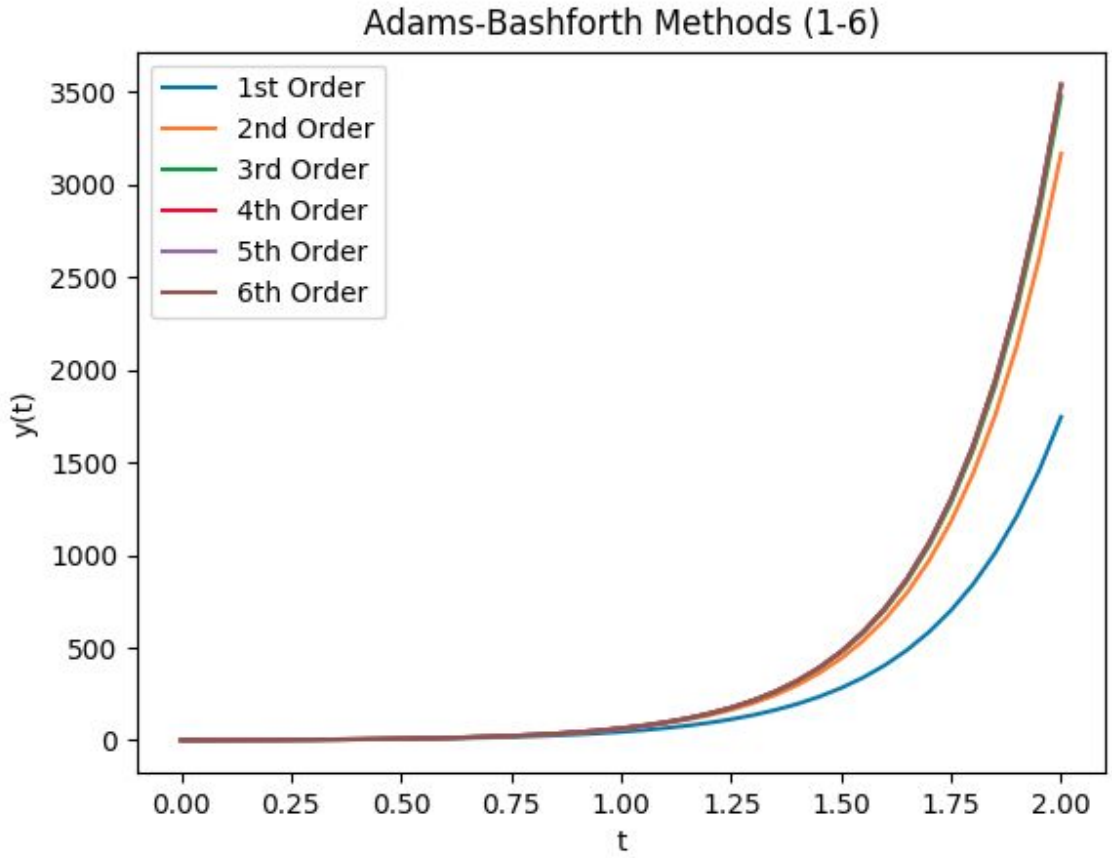


Ordem 6

```
Running Adams-Bashforth of order 6 for 35 iterations
y(0.0000) = 1.0000000
y(0.0500) = 1.2754125
y(0.1000) = 1.6090338
y(0.1500) = 2.0137514
y(0.2000) = 2.5053060
y(0.2500) = 3.1029232
y(0.3000) = 3.8300847
y(0.3500) = 4.7154719
y(0.4000) = 5.7941164
y(0.4500) = 7.1088049
y(0.5000) = 8.7117975
y(0.5500) = 10.6669249
y(0.6000) = 13.0521496
y(0.6500) = 15.9626949
y(0.7000) = 19.5148667
y(0.7500) = 23.8507212
y(0.8000) = 29.1437652
y(0.8500) = 35.6059204
y(0.9000) = 43.4960278
y(0.9500) = 53.1302357
y(1.0000) = 64.8946872
y(1.0500) = 79.2610182
y(1.1000) = 96.8052840
y(1.1500) = 118.2310785
y(1.2000) = 144.3977714
y(1.2500) = 176.3549964
y(1.3000) = 215.3847760
y(1.3500) = 263.0529721
y(1.4000) = 321.2721280
y(1.4500) = 392.3782239
y(1.5000) = 479.2244250
y(1.5500) = 585.2955872
y(1.6000) = 714.8481120
y(1.6500) = 873.0807678
y(1.7000) = 1066.3433287
y(1.7500) = 1302.3914075
y(1.8000) = 1590.6977078
y(1.8500) = 1942.8321873
y(1.9000) = 2372.9263899
y(1.9500) = 2898.2405800
y(2.0000) = 3539.8564403
```



Todos os métodos



Adams-Moulton

O método de Adams-Moulton é uma variação do método de Adams-Bashforth com a seguinte modificação: ele é um método implícito, exatamente como o Euler Inverso. Então o polinômio de interpolação utiliza o ponto que ainda vai ser descoberto no cálculo. Dessa forma, um método de ordem n precisa de apenas n - 1 pontos iniciais (n > 1).

Isso fica mais claro com as fórmulas:

$$y_n = y_{n-1} + hf(t_n, y_n), \qquad \text{(This is the backward Euler method)}$$

$$y_{n+1} = y_n + \frac{1}{2}h\left(f(t_{n+1}, y_{n+1}) + f(t_n, y_n)\right), \qquad \text{(This is the trapezoidal rule)}$$

$$y_{n+2} = y_{n+1} + h\left(\frac{5}{12}f(t_{n+2}, y_{n+2}) + \frac{2}{3}f(t_{n+1}, y_{n+1}) - \frac{1}{12}f(t_n, y_n)\right),$$

$$y_{n+3} = y_{n+2} + h\left(\frac{3}{8}f(t_{n+3}, y_{n+3}) + \frac{19}{24}f(t_{n+2}, y_{n+2}) - \frac{5}{24}f(t_{n+1}, y_{n+1}) + \frac{1}{24}f(t_n, y_n)\right),$$

$$y_{n+4} = y_{n+3} + h\left(\frac{251}{720}f(t_{n+4}, y_{n+4}) + \frac{646}{720}f(t_{n+3}, y_{n+3}) - \frac{264}{720}f(t_{n+2}, y_{n+2}) + \frac{106}{720}f(t_{n+1}, y_{n+1}) - \frac{19}{720}f(t_n, y_n)\right)$$

$$y_{n+5} = y_{n+4} + \frac{h}{1440}(475f_{n+5} + 1427f_{n+4} - 798f_{n+3} + 482f_{n+2} - 173f_{n+1} + 27f_n$$

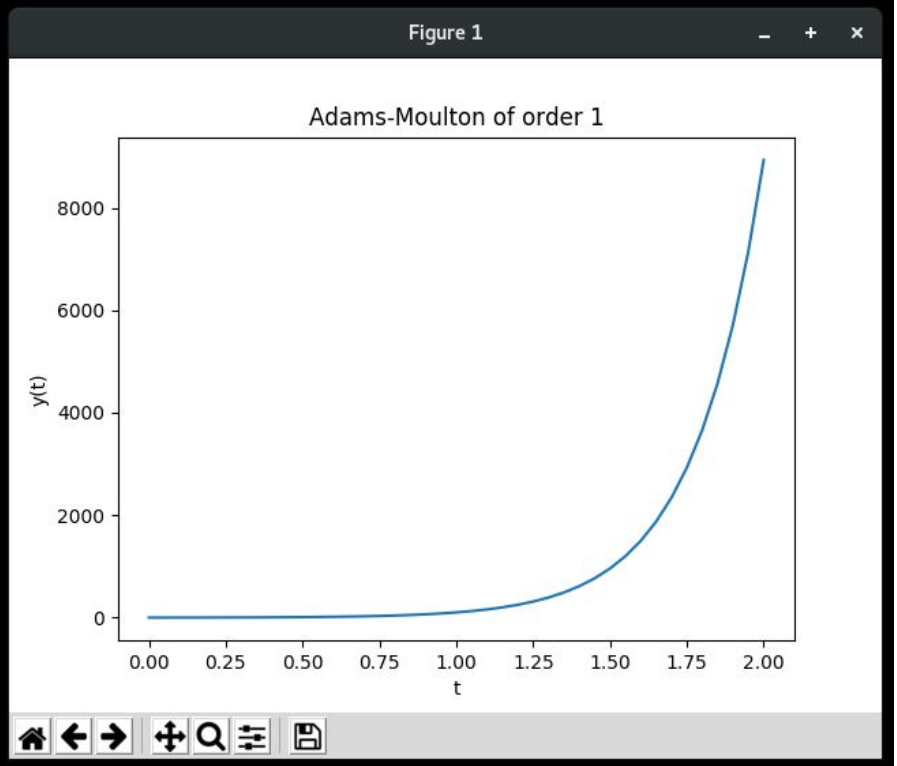
Onde:

$$f_{n+i} = f(t_{n+i}, y_{n+i})$$

Segue as implementações para cada ordem do método:

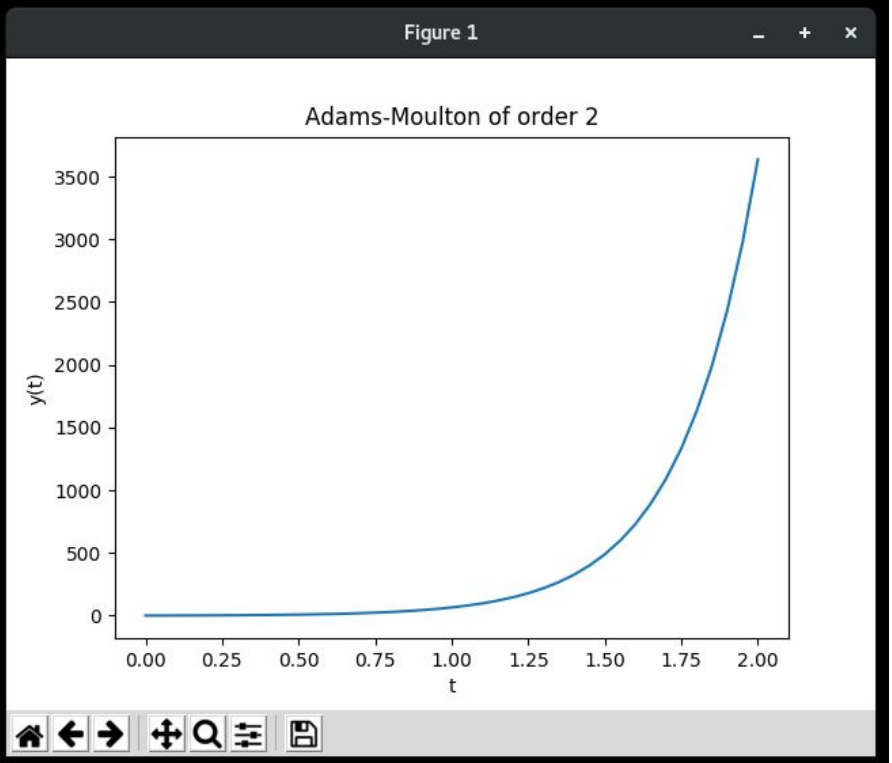
Ordem 1

```
Running Adams-Moulton of order 1 for 40 iterations
y(0.0000) = 1.0000000
y(0.0500) = 1.3093750
y(0.1000) = 1.6929687
y(0.1500) = 2.1693359
y(0.2000) = 2.7616699
y(0.2500) = 3.4989624
y(0.3000) = 4.4174530
y(0.3500) = 5.5624413
y(0.4000) = 6.9905516
y(0.4500) = 8.7725645
y(0.5000) = 10.9969556
y(0.5500) = 13.7743195
y(0.6000) = 17.2428993
y(0.6500) = 21.5754992
y(0.7000) = 26.9881240
y(0.7500) = 33.7507799
y(0.8000) = 42.2009749
y(0.8500) = 52.7605937
y(0.9000) = 65.9569921
y(0.9500) = 82.4493651
y(1.0000) = 103.0617064
y(1.0500) = 128.8240080
y(1.1000) = 161.0237600
y(1.1500) = 201.2703250
y(1.2000) = 251.5754062
y(1.2500) = 314.4536328
y(1.3000) = 393.0482910
y(1.3500) = 491.2884887
y(1.4000) = 614.0856109
y(1.4500) = 767.5788886
y(1.5000) = 959.4423607
y(1.5500) = 1199.2685759
y(1.6000) = 1499.0482199
y(1.6500) = 1873.7696499
y(1.7000) = 2342.1683124
y(1.7500) = 2927.6635155
y(1.8000) = 3659.5293943
y(1.8500) = 4574.3586179
y(1.9000) = 5717.8920224
y(1.9500) = 7147.3056530
y(2.0000) = 8934.0695662
```



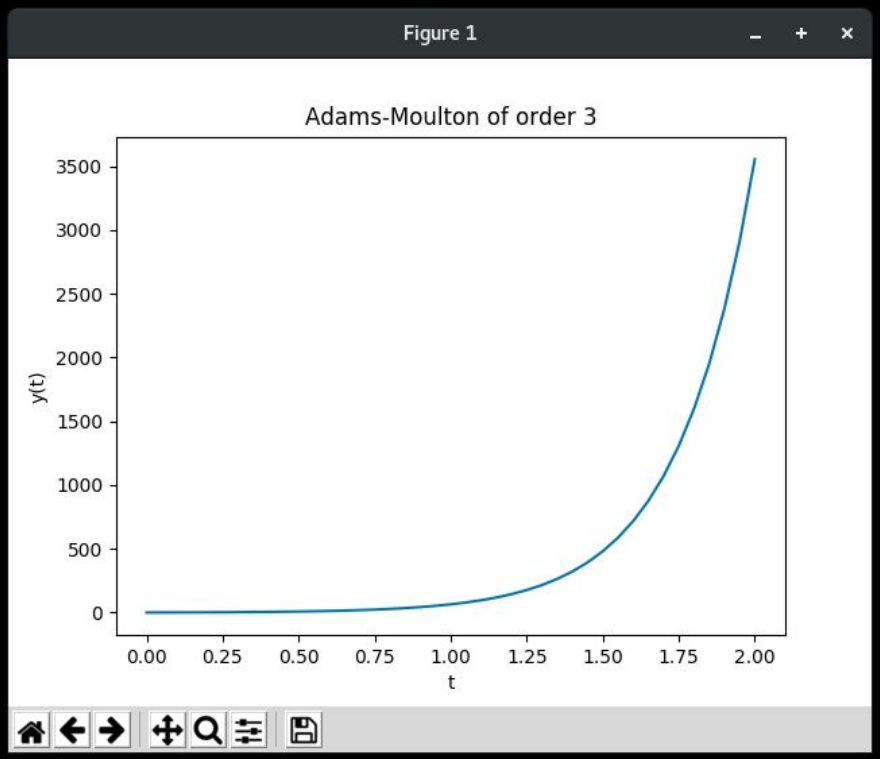
Ordem 2

```
Running Adams-Moulton of order 2 for 40 iterations
y(0.0000) = 1.0000000
y(0.0500) = 1.2763889
y(0.1000) = 1.6114198
y(0.1500) = 2.0181241
y(0.2000) = 2.5124295
y(0.2500) = 3.1138027
y(0.3000) = 3.8460367
y(0.3500) = 4.7382115
y(0.4000) = 5.8258696
y(0.4500) = 7.1524517
y(0.5000) = 8.7710521
y(0.5500) = 10.7465637
y(0.6000) = 13.1583001
y(0.6500) = 16.1032001
y(0.7000) = 19.6997445
y(0.7500) = 24.0927433
y(0.8000) = 29.4591863
y(0.8500) = 36.0153944
y(0.9000) = 44.0257598
y(0.9500) = 53.8134286
y(1.0000) = 65.7733572
y(1.0500) = 80.3882699
y(1.1000) = 98.2481632
y(1.1500) = 120.0741440
y(1.2000) = 146.7475648
y(1.2500) = 179.3456348
y(1.3000) = 219.1849425
y(1.3500) = 267.8746520
y(1.4000) = 327.3815191
y(1.4500) = 400.1093567
y(1.5000) = 488.9961582
y(1.5500) = 597.6328044
y(1.6000) = 730.4081498
y(1.6500) = 892.6863498
y(1.7000) = 1091.0235942
y(1.7500) = 1333.4330040
y(1.8000) = 1629.7083938
y(1.8500) = 1991.8199813
y(1.9000) = 2434.3980327
y(1.9500) = 2975.3239845
y(2.0000) = 3636.4529255
```



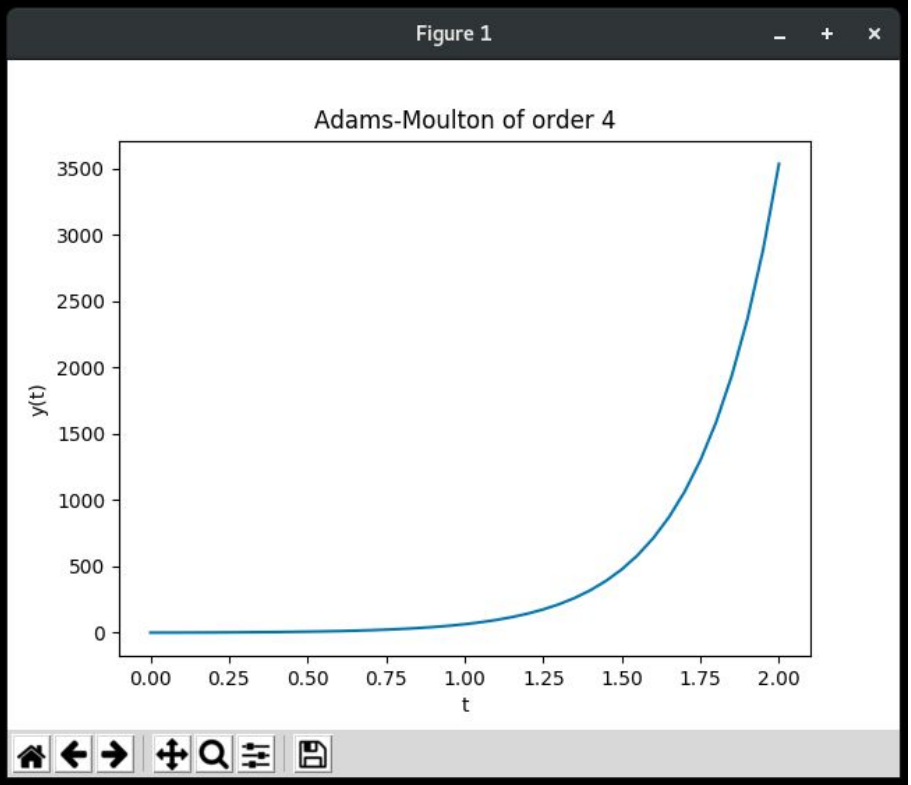
Ordem 3

```
Running Adams-Moulton of order 3 for 38 iterations
y(0.0000) = 1.0000000
y(0.1000) = 1.6090338
y(0.1500) = 2.0186691
y(0.2000) = 2.5115539
y(0.2500) = 3.1107636
y(0.3000) = 3.8399158
y(0.3500) = 4.7277911
y(0.4000) = 5.8095433
y(0.4500) = 7.1281118
y(0.5000) = 8.7359466
y(0.5500) = 10.6971137
y(0.6000) = 13.0898688
y(0.6500) = 16.0097994
y(0.7000) = 19.5736635
y(0.7500) = 23.9240785
y(0.8000) = 29.2352486
y(0.8500) = 35.7199605
y(0.9000) = 43.6381284
y(0.9500) = 53.3072322
y(1.0000) = 65.1150666
y(1.0500) = 79.5353145
y(1.1000) = 97.1465695
y(1.1500) = 118.6555711
y(1.2000) = 144.9255866
y(1.2500) = 177.0110785
y(1.3000) = 216.2000500
y(1.3500) = 264.0657695
y(1.4000) = 322.5299505
y(1.4500) = 393.9399248
y(1.5000) = 481.1629079
y(1.5500) = 587.7011421
y(1.6000) = 717.8325410
y(1.6500) = 876.7824844
y(1.7000) = 1070.9336618
y(1.7500) = 1308.0823912
y(1.8000) = 1597.7517080
y(1.8500) = 1951.5737956
y(1.9000) = 2383.7571162
y(1.9500) = 2911.6570019
y(2.0000) = 3556.4726184
```



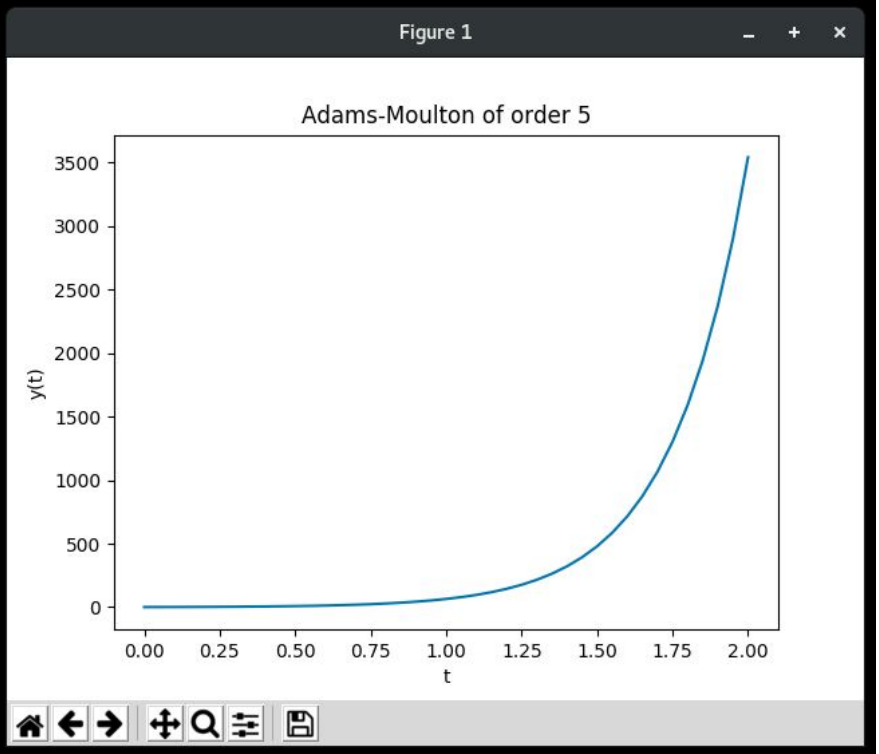
Ordem 4

```
Running Adams-Moulton of order 4 for 37 iterations
y(0.0000) = 1.0000000
y(0.1000) = 1.6090338
y(0.1500) = 2.0137514
y(0.2000) = 2.5029620
y(0.2500) = 3.1000179
y(0.3000) = 3.8265895
y(0.3500) = 4.7112505
y(0.4000) = 5.7890171
y(0.4500) = 7.1026461
y(0.5000) = 8.7043601
y(0.5500) = 10.6579445
y(0.6000) = 13.0413074
y(0.6500) = 15.9496070
y(0.7000) = 19.4990700
y(0.7500) = 23.8316577
y(0.8000) = 29.1207629
y(0.8500) = 35.5781696
y(0.9000) = 43.4625533
y(0.9500) = 53.0898633
y(1.0000) = 64.8460036
y(1.0500) = 79.2023219
y(1.1000) = 96.7345278
y(1.1500) = 118.1457997
y(1.2000) = 144.2950075
y(1.2500) = 176.2311854
y(1.3000) = 215.2356354
y(1.3500) = 262.8733550
y(1.4000) = 321.0558500
y(1.4500) = 392.1178560
y(1.5000) = 478.9110458
y(1.5500) = 584.9184859
y(1.6000) = 714.3944340
y(1.6500) = 872.5350895
y(1.7000) = 1065.6871510
y(1.7500) = 1301.6025493
y(1.8000) = 1589.7495828
y(1.8500) = 1941.6929407
y(1.9000) = 2371.5578684
y(1.9500) = 2896.5971043
y(2.0000) = 3537.8833432
```



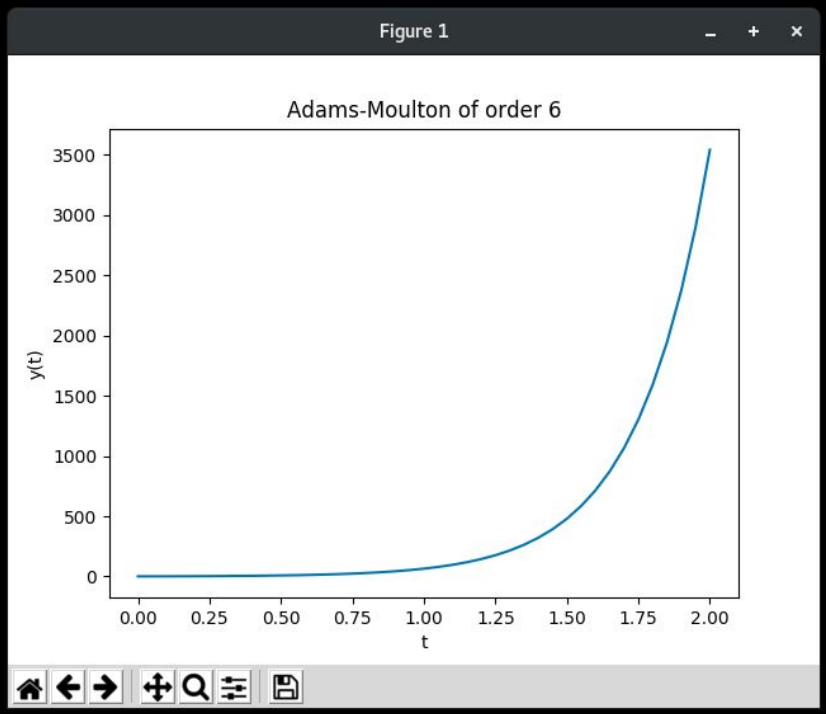
Ordem 5

```
Running Adams-Moulton of order 5 for 36 iterations
y(0.0000) = 1.0000000
y(0.1000) = 1.6090338
y(0.1500) = 2.0137514
y(0.2000) = 2.5053060
y(0.2500) = 3.1044249
y(0.3000) = 3.8320016
y(0.3500) = 4.7177998
y(0.4000) = 5.7969835
y(0.4500) = 7.1123324
y(0.5000) = 8.7161357
y(0.5500) = 10.6722599
y(0.6000) = 13.0587101
y(0.6500) = 15.9707622
y(0.7000) = 19.5247864
y(0.7500) = 23.8629180
y(0.8000) = 29.1587612
y(0.8500) = 35.6243573
y(0.9000) = 43.5186941
y(0.9500) = 53.1581004
y(1.0000) = 64.9289413
y(1.0500) = 79.3031248
y(1.1000) = 96.8570412
y(1.1500) = 118.2946957
y(1.2000) = 144.4759631
y(1.2500) = 176.4510978
y(1.3000) = 215.5028848
y(1.3500) = 263.1981224
y(1.4000) = 321.4505044
y(1.4500) = 392.5974237
y(1.5000) = 479.4937815
y(1.5500) = 585.6265648
y(1.6000) = 715.2547934
y(1.6500) = 873.5804507
y(1.7000) = 1066.9572596
y(1.7500) = 1303.1456819
y(1.8000) = 1591.6243761
y(1.8500) = 1943.9706127
y(1.9000) = 2374.3249150
y(1.9500) = 2899.9585737
y(2.0000) = 3541.9668104
```



Ordem 6

```
Running Adams-Moulton of order 6 for 35 iterations
y(0.0000) = 1.0000000
y(0.1000) = 1.6090338
y(0.1500) = 2.0137514
y(0.2000) = 2.5053060
y(0.2500) = 3.1029232
y(0.3000) = 3.8290390
y(0.3500) = 4.7141426
y(0.4000) = 5.7925498
y(0.4500) = 7.1068961
y(0.5000) = 8.7094884
y(0.5500) = 10.6641338
y(0.6000) = 13.0487740
y(0.6500) = 15.9586132
y(0.7000) = 19.5099318
y(0.7500) = 23.8447552
y(0.8000) = 29.1365536
y(0.8500) = 35.5972040
y(0.9000) = 43.4854937
y(0.9500) = 53.1175063
y(1.0000) = 64.8793071
y(1.0500) = 79.2424373
y(1.1000) = 96.7828389
y(1.1500) = 118.2039691
y(1.2000) = 144.3650324
y(1.2500) = 176.3154640
y(1.3000) = 215.3370468
y(1.3500) = 262.9953543
y(1.4000) = 321.2025827
y(1.4500) = 392.2942938
y(1.5000) = 479.1231495
y(1.5500) = 585.1733997
y(1.6000) = 714.7007172
y(1.6500) = 872.9029933
y(1.7000) = 1066.1289473
y(1.7500) = 1302.1329237
y(1.8000) = 1590.3861016
y(1.8500) = 1942.4566066
y(1.9000) = 2372.4737812
y(1.9500) = 2897.6952454
y(2.0000) = 3539.1995066
```



Todos os métodos

Pode parecer existir apenas das funções mas todas estão plotadas, o que acontece é que os valores são muito próximos, como pode ser visto com as tabelas.