Universidade Federal de Pernambuco
Centro de Informática

Métodos Numéricos e Computacionais

Aluna: Izabella Maria Cavalcanti Melo
Professor: Ricardo Martins

Outubro
2018

Universidade Federal de Pernambuco
Centro de Informática

Métodos Numéricos e Computacionais

Implementações e Análise dos Métodos

Outubro
2018

Conteúdo

1 Introdução

Equações diferenciais podem ser resolvidas analiticamente como forma de obter a sua solução exata, mas nem sempre isso é possível, principalmente em problemas não-lineares. Para isso, surgiram os Métodos Numéricos, técnicas de resolução de equações diferenciais que dão resultados aproximados, mas satisfatórios.

Todos os métodos irão tentar solucionar equações diferenciais ordinárias com um valor inicial dado, ou seja:

$$y' = f(t, y)$$

Com a condição inicial:

$$y(t_0) = y_0$$

Neste relatório serão analisadas as implementações dos algoritmos feitas para o projeto da cadeira de Métodos Numéricos e Computacionais, do Centro de Informátida - UFPE, bem como os métodos em si e seus respectivos erros.

2 Métodos

2.1 Euler

Euler é o método numérico mais simples e possui a seguinte fórmula:

$$y_{n+1} = y_n + h * f(t_n, y_n) \tag{1}$$
$$n = 0, 1, 2, ...$$

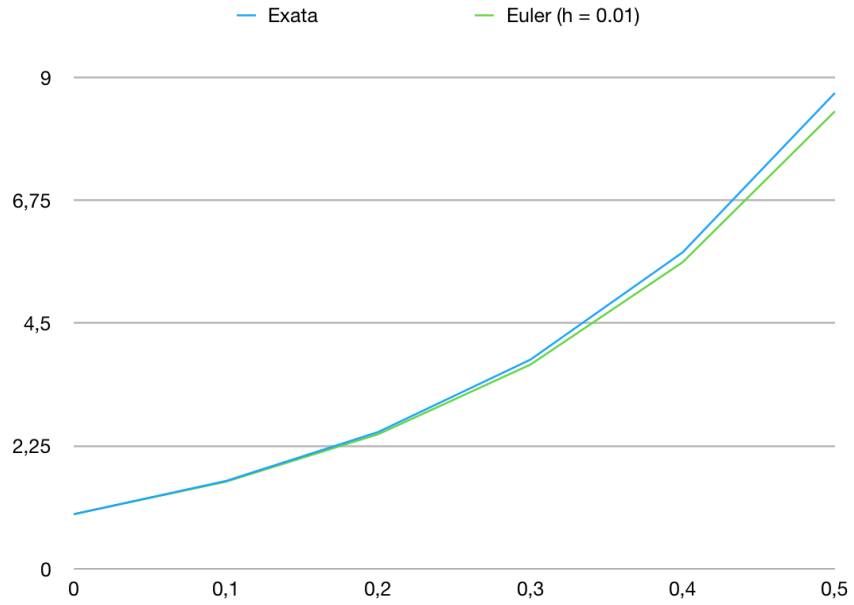
A ideia de Euler é utilizar a reta tangente ao ponto inicial como forma de descobrir os pontos seguintes. Observe que o método consiste em calcular repetidamente a Eq. (1), obtendo, assim, os valores aproximados de y.

A implementação realizada no projeto foi feita a partir no seguinte algoritmo:

Algorithm 1 Método de Euler

```
1: function EULER(y, t, h, n, f(t, y))
2:     for i ← 0 to n do
3:         y ← y + h * f(t, y)
4:         t ← t + h
5:         escreva t e y
6:     end for
7: end function
```

No gráfico abaixo pode ser visto a diferença entre o resultado exato da equação $y' = 1 - t + 4 * y$ e o obtido pela implementação do algoritmo acima. A diferença entre os dois resultados é o erro. Euler possui erro de ordem h^2:



É fácil perceber que quanto menor o intervalo $h = t_n - t_{n-1}$ mais próxima a solução dada por Euler ficará da solução exata.

2.2 Euler Inverso

Euler Inverso é uma variante da fórmula de Euler cuja equação implicita pode ser vista abaixo:

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

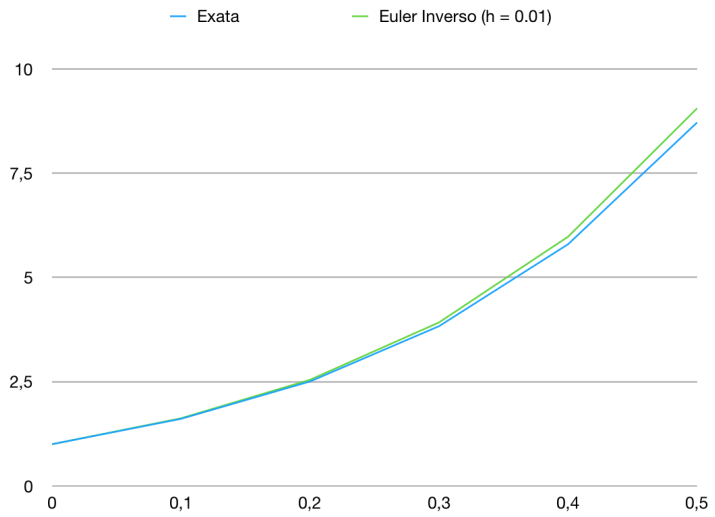
A resolução da equação implicita sugere a utilização de um método de previsão, ou seja, é normal utilizar Euler para prever o valor de y_{n+1} e depois usar na fórmula de Euler Inverso, como pode ser visto no pseudo-código abaixo:

Algorithm 2 Método de Euler Inverso

```
1: function BACKWARD-EULER(y, t, h, n, f(t, y))
2:     for i ← 0 to n do
3:         y_euler ← y + h * f(t, y)
4:         t ← t + h
5:         y ← y + h * f(t, y_euler)
6:         escreva t e y
7:     end for
8: end function
```

Apesar de, em certos casos, possuir maiores erros que Euler e um algoritmo mais complicado, Euler Inverso é muito útil para alguns tipos de equações diferenciais.

Abaixo é possível ver os valores obtidos para y em relação aos valores exatos, no gráfico.



2.3 Euler Aprimorado

Euler Aprimorado, como dito pelo próprio nome, surgiu como uma ferramenta para tentar tornar os valores obtidos por Euler mais próximos da solução exata já que tanto Euler quanto Euler Inverso precisam de valores de h muito pequenos para apresentar resultados satisfatórios em algumas ocasiões.

Para isso a fórmula tira a média dos valores de $f(t_n, y_n)$ e $f(t_{n+1}, y_{n+1})$, como pode ser visto abaixo:

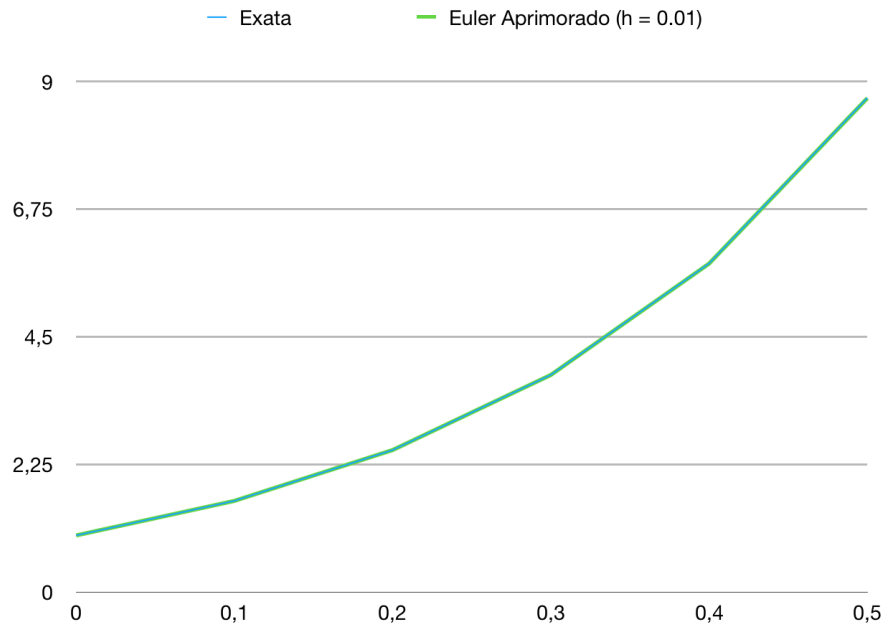
$$y_{n+1} = y_n + \frac{h * (f(t_n, y_n) + f(t_{n+1}, y_{n+1})}{2} \tag{3}$$

Assim como em Euler Inverso, a Eq. (3) também é implicita e pede o método da previsão por Euler para ser resolvida no seguinte algoritmo:

Algorithm 3 Método de Euler Aprimorado

```
1: function BEST-EULER(y, t, h, n, f(t, y))
2:     for i ← 0 to n do
3:         f_n ← y + h * f(t, y)
4:
5:         y_euler ← y + h * f(t, y)
6:         t ← t + h
7:         f_{n+1} ← y + h * f(t, y_euler)
8:
9:         y ← y + (h/2) * (f_n + f_{n+1})
10:        escreva t e y
11:    end for
12: end function
```

É provado que o erro de Euler Aprimorado é da ordem de h^3. Um exemplo da sua precisão pode ser vista no gráfico abaixo, onde, para os valores que estávamos acostumados, o método foi tão preciso que não é possível sequer diferenciá-lo da solução exata:



Por isso, para se ter uma noção melhor da diferença entre os resultados, segue a tabela:

h	Exata	Euler Aprimorado
0	1	1
0,1	1,6090418	1,608858451759808
0,2	2,5053299	2,504782749154446
0,3	3,8301388	3,828914635497995
0,4	5,7942260	5,791791053877201
0,5	8,7120041	8,707463704118913

2.4 Runge-Kutta

O método de Runge-Kutta é um pouco mais complexo que os métodos de euler, mas possui um erro bem menor, da ordem de h^5 e, por isso, é muito utilizado para tratar problemas de maneira eficiente.

A fórmula de Runge-Kutta é:

$$y_{n+1} = y_n + \frac{h * (k_{n1} + 2 * k_{n2} + 2k_{n3} + k_{n4})}{6} \tag{4}$$

onde

$$k_{n1} = f(t_n, y_n)$$
$$k_{n1} = f(t_n, y_n)$$
$$k_{n2} = f(t_n + \frac{h}{2}, y_n + \frac{h * k_{n1}}{2})$$
$$k_{n3} = f(t_n + \frac{h}{2}, y_n + \frac{h * k_{n2}}{2})$$
$$k_{n4} = f(t_n + h, y_n + h * k_{n3})$$

Como esperado, a tabela abaixo mostra como Runge-Kutta se aproxima ainda mais da solução exata que Euler Aprimorado:

h	Exata	Runge-Kutta
0	1	1
0,1	1,6090418	1,609041813827218
0,2	2,5053299	2,50532980895851
0,3	3,8301388	3,830138748125474
0,4	5,7942260	5,794225809784991
0,5	8,7120041	8,712003755369151

O algoritmo de Runge-kutta é mais complicado do que os anteriores, mas ainda assim pode ser implementado sem muita dificuldade a partir do pseudo-código abaixo:

Algorithm 4 Método de Runge-kutta

```
1: function RUNGE-KUTTA(y, t, h, n, f(t, y))
2:     for i ← 0 to n do
3:         k1 ← f(t, y)
4:         k2 ← f(t + 0.5 * h, y + 0.5 * h * k1)
5:         k3 ← f(t + 0.5 * h, y + 0.5 * h * k2)
6:         k1 ← f(t + h, y + h * k3)
7:         y ← y + (h/6) * (k1 + k2 + k3 + k4)
8:         escreva t e y
9:     end for
10: end function
```

As diferenças entre esse método e a solução exata nessa equação só começam a ficar evidentes em passos mais altos ou com intervalos de h maiores. Isso tudo faz de Runge-kutta um bom método para resolução de muitos problemas com equações diferenciais ordinárias, inclusive o que estava sendo usado de exemplo nos gráficos anteriores, mas ainda era necessário tornar o cálculo mais preciso em problemas maiores.

2.5 Métodos de Adams

Pensando em diminuir os erros que os métodos ainda possuiam em casos mais específicos de equações diferenciais ordinárias e reduzir o custo computacional de métodos como Runge-kutta, surgiram os métodos de múltiplos passos que, diferente dos de passo único, utilizam pontos anteriores ao último para encontrar o y_{n+1}

O método de Adams é um método de passo múltiplo que surgiu a partir da ideia de aproximar y' de um polinômio de grau k. A ordem do método será definida como $k + 1$.

É importante observar que a fórmula de Adams-Bashforth de primeira ordem (aproximação do polinômio de grau zero) resulta no próprio método de Euler. Conforme se aumenta o grau do polinômio de aproximação, o erro de Adams-Bashforth diminui, mas o cálculo necessitará de mais pontos anteriores.

Abaixo segue a tabela de coeficientes de Adams-Bashforth de 2 a 8:

k	β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8
1	1							
2	$\frac{3}{2}$	$-\frac{1}{2}$						
3	$\frac{23}{12}$	$-\frac{4}{3}$	$\frac{5}{12}$					
4	$\frac{55}{24}$	$-\frac{59}{24}$	$\frac{37}{24}$	$-\frac{3}{8}$				
5	$\frac{1901}{720}$	$-\frac{1387}{360}$	$\frac{109}{30}$	$-\frac{637}{360}$	$\frac{251}{720}$			
6	$\frac{4277}{1440}$	$-\frac{2641}{480}$	$\frac{4991}{720}$	$-\frac{3649}{720}$	$\frac{959}{480}$	$-\frac{95}{288}$		
7	$\frac{198721}{60480}$	$-\frac{18637}{2520}$	$\frac{235183}{20160}$	$-\frac{10754}{945}$	$\frac{135713}{20160}$	$-\frac{5603}{2520}$	$\frac{19087}{60480}$	
8	$\frac{16083}{4480}$	$-\frac{1152169}{120960}$	$\frac{242653}{13440}$	$-\frac{296053}{13440}$	$\frac{2102243}{120960}$	$-\frac{115747}{13440}$	$\frac{32863}{13440}$	$-\frac{5257}{17280}$

Da mesma ideia do Adams-Bashforth surgiu o Adams-Moulton, que é apenas uma variação da dedução das fórmulas e gera fórmulas implícitas. O Adams-Moulton de primeira ordem é o método de Euler Inverso.

Abaixo segue a tabela de coeficientes de Adams-Moulton de 2 a 8:

k	β_0	β_1	β_2	β_3	β_4	β_5	β_6	β_7
0	1							
1	$\frac{1}{2}$	$\frac{1}{2}$						
2	$\frac{5}{12}$	$\frac{2}{3}$	$-\frac{1}{12}$					
3	$\frac{3}{8}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$				
4	$\frac{251}{720}$	$\frac{323}{360}$	$-\frac{11}{30}$	$\frac{53}{360}$	$-\frac{19}{720}$			
5	$\frac{95}{288}$	$\frac{1427}{1440}$	$-\frac{133}{240}$	$\frac{241}{720}$	$-\frac{173}{1440}$	$\frac{3}{160}$		
6	$\frac{19087}{60480}$	$\frac{2713}{2520}$	$-\frac{15487}{20160}$	$\frac{586}{945}$	$-\frac{6737}{20160}$	$\frac{263}{2520}$	$-\frac{863}{60480}$	
7	$\frac{5257}{17280}$	$\frac{139849}{120960}$	$-\frac{4511}{4480}$	$\frac{123133}{120960}$	$-\frac{88547}{120960}$	$\frac{1537}{4480}$	$-\frac{11351}{120960}$	$\frac{275}{24192}$

Para prever y_{n+1} o algoritmo utilizado foi o Euler Inverso visando manter a precisão esperada.

2.6 Fórmula Inversa

Ainda falando de métodos de múltiplos passos, as fórmulas inversas de diferenciação surgiram a partir da ideia de aproximar o polinômio de grau k da solução exata $\phi(t)$ ao contrário dos métodos de Adams que aproximam o polinômio de $\phi'(t)$.

Da mesma forma de Adams-Moulton, a fórmula inversa de primeira ordem é o método de Euler Inverso.

Order	Formula
1	$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{f}_{n+1}$
2	$\boldsymbol{y}_{n+2} - \frac{4}{3}\boldsymbol{y}_{n+1} + \frac{1}{3}\boldsymbol{y}_n = \frac{2h}{3}\boldsymbol{f}_{n+2}$
3	$\boldsymbol{y}_{n+3} - \frac{18}{11}\boldsymbol{y}_{n+2} + \frac{9}{11}\boldsymbol{y}_{n+1} - \frac{2}{11}\boldsymbol{y}_n = \frac{6h}{11}\boldsymbol{f}_{n+3}$
4	$\boldsymbol{y}_{n+4} - \frac{48}{25}\boldsymbol{y}_{n+3} + \frac{36}{25}\boldsymbol{y}_{n+2} - \frac{16}{25}\boldsymbol{y}_{n+1} + \frac{3}{25}\boldsymbol{y}_n = \frac{12h}{25}\boldsymbol{f}_{n+4}$
5	$\boldsymbol{y}_{n+5} - \frac{300}{137}\boldsymbol{y}_{n+4} + \frac{300}{137}\boldsymbol{y}_{n+3} - \frac{200}{137}\boldsymbol{y}_{n+2} + \frac{75}{137}\boldsymbol{y}_{n+1} - \frac{12}{137}\boldsymbol{y}_n = \frac{60h}{137}\boldsymbol{f}_{n+5}$
6	$\boldsymbol{y}_{n+6} - \frac{360}{147}\boldsymbol{y}_{n+5} + \frac{450}{147}\boldsymbol{y}_{n+4} - \frac{400}{147}\boldsymbol{y}_{n+3} + \frac{225}{147}\boldsymbol{y}_{n+2} - \frac{72}{147}\boldsymbol{y}_{n+1} + \frac{10}{147}\boldsymbol{y}_n = \frac{60h}{147}\boldsymbol{f}_{n+6}$

A fórmula de cada ordem do método pode ser encontrada isolando o y_{n+max} e passando todos os outros valores para o outro lado da igualdade. Observa-se que esse método também é implicito e na implementação, visando manter um erro aceitável, foi utilizado o método de Euler Inverso para previsão do y_{n+max}

3 Conclusão

Como dito inicialmente, todos os métodos acima tem como principal objetivo fornecer a melhor aproximação da solucão exata para a resolução de equações diferenciais ordinárias, principalmente aquelas cuja solução exata é muito difícil de ser calculada.

Podemos dividir tais métodos em métodos de passo único e métodos de múltiplos passo. Os métodos de passo único: Euler e variações e Runge-kutta chegam a oferecer boas aproximações para o problema do valor inicial, mas exigem maior custo computacional para isso. Runge-kutta, por exemplo, ofereceu valores muito próximos à solução exata no exemplo dado, mas necessita de quatro cáculos de valores de f em cada passo, enquanto o método de Adams-Bashforth de quarta ordem necessita de apenas um.

Assim, todos os métodos oferecem boas aproximações para o problema do valor inicial e o método escolhido deve ser o que mais se enquadra em cada ocasião, levando em consideração desde custo computacional até os resultados obtidos.