Projeto IF 816 CIn-UFPE

Parte 1

Vinícius Serra Silva de Melo

20/10/2018

Sumário

1. Considerações inicias 3

2. Métodos de Passo Simples
   2.1 Método de Euler 4
   2.2 Método de Euler Inverso 4
   2.2. Método de Euler Aprimorado 5
   2.3. Método de Runge – Kutta 6

3. Métodos de passo múltiplos
   3.1 Método de Adams – Bashforth 7
   3.2 Método de Adamas – Multon 8
   3.3 Método da diferenciação inversa 9

4. Comparação entre os métodos 10

5. Conclusão 14

1. Considerações iniciais

O problema que devemos resolver se resume a encontrar soluções aproximadas para equações diferencias ordinárias da forma $\frac{dy}{dt} = f(t,\ y)\ (1)$. Se tomarmos a integral de ambos os lados da equação (1), chegamos a uma nova equação da forma $Y_{n+1} = Yn + \int_{t_n}^{t_{n+1}} f\big(t, y(t)\big)dt$ (2). Durante os métodos estudados nesse texto, normalmente iremos nos basear na equação (2). Veremos que os primeiros métodos, conhecidos como métodos de passo simples, aproximam a integral do lado direito da equação de forma geométrica, enquanto nos métodos de passo múltiplos, são utilizados polinômios para aproximar tanto a derivada, como a própria solução do problema.

Durante o curso do texto, utilizaremos como exemplo o problema de valor inicial dado por $\frac{dy}{dt} = 1 - t + 4y,\ y(0) = 0\ (3)$. A equação (3) possui solução analítica exata, o que facilita nossa análise das soluções e dos seus erros. A solução exata é dada por $Y(t) = \frac{1}{16}(4t + 3e^{4t} - 3)\ (4)$. Abaixo é possível analisar uma tabela com os valores de Y para t variando no intervalo (0, 1) com um passo entre pontos de h = 0.1, a mesma tabela será utilizada nos métodos apresentados nas próximas sessões.

T	Y
0.0	0.000000
0.1	0.117217
0.2	0.279789
0.3	0.510022
0.4	0.841194
0.5	1.322948
0.6	2.029346
0.7	3.070871
0.8	4.612349
0.9	6.899669
1.0	10.299653

2. Métodos de passo simples

2.0 Introdução

Nessa primeira sessão, serão estudados os métodos de passo simples, que incluem os métodos de Euler, Euler Inverso, Euler Aprimorado e Runge – Kutta.

Os métodos possuem essa denominação de passo simples, pois para aproximarmos a solução de Yn+1 na equação 1.2, usamos apenas o valor do passo anterior, ou seja, Yn. Os métodos apresentados abaixo procuram aproximações geométricas para a integral do lado direito da equação 1.2.

2.1 Método de Euler

O método de Euler utiliza a aproximação mais simples para a integral de 1.2. O valor é aproximado pelo retângulo de base h, e altura $f(t_n, y_n)$. Diante do exposto acima, podemos reescrever a equação 1.2 na forma $Y_{n+1} = Y_n + hf(t_n, y_n)$ (1). A partir da equação 2.1, é simples criar uma função em python para aproximar a solução da equação inicial. O método apresentado é conhecido como Método de Euler, e é a forma mais simples de aproximarmos numericamente soluções para equação diferenciais ordinárias.

```
def euler(self):
    ans = []
    ans.append([self.t0, self.y0])
    t, y = self.t0, self.y0
    for i in range(1, self.nsteps+1):
        y = y + self.h*self.f(t, y)
        t = t + self.h
        ans.append([t, y])

    return ans
```

2.2 Método de Euler Inverso

O método de Euler Inverso apresenta grande semelhança com o apresentado na sessão 2.1. A diferença ocorre na aproximação geométrica da integral de 1.2, aqui usamos um retângulo de base H e altura $f(t_{n+1}, y_{n+1})$. Usando essa aproximação para a integral, chegamos facilmente no seguinte resultado $Y_{n+1} = Y_n + hf(t_{n+1}, y_{n+1})$ (2). Chegamos agora em um impasse, pois a equação 2.2 é implícita em Y_{n+1}, pois precisamos do valor de $f(t_{n+1}, y_{n+1})$ para obetermos Y_{n+1}. Uma solução possível é resolver implicitamente a equação, entretanto dessa forma tornamos o programa mais lento. Outra possibilidade, que foi a utilizada na implementação do projeto, é a de aproximarmos inicialmente o valor de Y_{n+1} pelo método de Euler, apresentado acima. Dessa forma, chegamos em uma equação da forma $Y_{n+1} = Y_n + hf\big(t_{n+1}, Y_n + hf(t, y_n)\big)$ (3) . Novamente, observamos que escrever um programa em python que utilize a equação (3) para aproximar a solução de uma equação diferencial é uma tarefa simples.

```
def inverse_euler(self):
    ans = []
    ans.append([self.t0, self.y0])
    t, y = self.t0, self.y0
    for i in range(1, self.nsteps+1):
        k = y + self.h*self.f(t, y)
        y = y + self.h*self.f(t + self.h, k)
        t = t + self.h
        ans.append([t, y])

    return ans
```

2.3 Método de Euler Aprimorado.

O método de Euler Aprimorado pode ser entendido como uma combinação de ambos os métodos apresentados acima. Ele utiliza não mais um retângulo para aproximar a integral inicial, mas agora um trapézio retângulo de altura H, base menor $f(t_n, y_n)$ e base maior $f(t_{n+1}, y_{n+1})$. Chegamos então novamente em uma equação implícita da forma $Y_{n+1} = Y_n + \frac{h}{2}\big(f(t_n, y_n) + f(t_{n+1}, y_{n+1})\big)(4)$. Novamente, a resolução de uma equação implícita nos causa os problemas já explicados acima, portanto faremos uma nova aproximação usando o método de Euler, o que nos leva a seguinte equação $Y_{n+1} = Y_n + \frac{h}{2}\Big(f(t_n, y_n) + f\big(t_{n+1}, Y_n + hf(t, y_n)\big)\Big)$ (5). Novamente, a equação simples pode ser facilmente traduzida para um programa em python.

```
def improved_euler(self):
    ans = []
    ans.append([self.t0, self.y0])
    t, y = self.t0, self.y0
    for i in range(1, self.nsteps+1):
        k = y + self.h*self.f(t, y)
        y = y + 0.5*self.h*(self.f(t + self.h, k) + self.f(t, y))
        t = t + self.h
        ans.append([t, y])

    return ans
```

2.4 Método de Runge-Kutta

O método de Runge-Kutta apresenta uma abordagem um pouco diferente para resolver o problema de aproximar a integral incialmente apresentada. É utilizado agora o valor dado por $\frac{h}{6}(k1 + 2k2 + 2k3 + k4),\ k1 = f(t, y),\ k2 = f(t + 0.5 * h, y + 0.5 * h * k1),\ k3 = f(t + 0.5 * h, y + 0.5 * h * k2),\ k4 = f(t + h, y + h * k3)\ (6)$. A equação (6) pode ser interpretada como uma média ponderada de várias áreas possíveis para aproximação da integral. Vemos também que a equação (6) é facilmente transformada em código, pois não é uma equação implícita.

```
def runge_kutta(self):
    ans = []
    t, y = self.t0, self.y0
    ans.append([t, y])
    for i in range(1, self.nsteps+1):
        k1 = f(t, y)
        k2 = f(t + 0.5*h, y + 0.5*h*k1)
        k3 = f(t + 0.5*h, y + 0.5*h*k2)
        k4 = f(t + h, y + h*k3)
        y = y + h*(k1 + 2*k2 + 2*k3 + k4)/6
        t = t + h
        ans.append([t, y])

    return ans
```

3. Métodos de passos múltiplos

3.0 Introdução

Nessa sessão, os métodos apresentados utilizam polinômios de grau variado para aproximar tanto o valor de $\frac{dy}{dt}$ (métodos de Adams-Bashforth e Adams-Multon), como a própria solução Y da equação diferencial inicial (método da fórmula inversa de diferenciação). Os coeficientes dos polinômios são determinados a partir de pontos iniciais, que podem ser gerados usando outros métodos já apresentados, como a partir de uma lista de pontos inicias. Geralmente, se utilizamos um polinômio de grau K para aproximar uma função, teremos K + 1 coeficientes a serem determinados, e precisaremos de K pontos iniciais para resolver um sistema linear e encontrarmos os coeficientes.

3.1 Método de Adams-Bashforth

Esse método utiliza um polinômio de grau K $P(t) = A_{k+1}t^k + \cdots + A_1 t + A_0$ para aproximar o valor de $\frac{dy}{dt}$. Como sabemos pela equação inicial, que $\frac{dy}{dt} = f(t, y)$, podemos criar um sistema linear com K+1 incógnitas (os coeficiente de P(t)) se soubermos de antemão K+1 pontos iniciais para o problema. Depois que os coeficientes são inteiramente determinados, fazemos a integral do polinômio e chegamos numa equação da forma

$$Y_{n+1} = Y_n + \frac{A_{k+1}}{k+1} * ({t_{n+1}}^{k+1} - {t_n}^{k+1}) + \ldots + \frac{A_1}{2} * ({t_{n+1}}^2 - {t_n}^2) + A_0(t_{n+1} - t_n)$$

que pode ser facilmente resolvida quando determinados os coeficientes.

Abaixo segue uma tabela com os valores dos coeficientes para alguns valores pequenos de K.

Coefficients and error constants for Adams–Bashforth methods

k	β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8	C
1	1								$-\frac{1}{2}$
2	$\frac{3}{2}$	$-\frac{1}{2}$							$\frac{5}{12}$
3	$\frac{23}{12}$	$-\frac{4}{3}$	$\frac{5}{12}$						$-\frac{3}{8}$
4	$\frac{55}{24}$	$-\frac{59}{24}$	$\frac{37}{24}$	$-\frac{3}{8}$					$\frac{251}{720}$
5	$\frac{1901}{720}$	$-\frac{1387}{360}$	$\frac{109}{30}$	$-\frac{637}{360}$	$\frac{251}{720}$				$-\frac{95}{288}$
6	$\frac{4277}{1440}$	$-\frac{2641}{480}$	$\frac{4991}{720}$	$-\frac{3649}{720}$	$\frac{959}{480}$	$-\frac{95}{288}$			$\frac{19087}{60480}$
7	$\frac{198721}{60480}$	$-\frac{18637}{2520}$	$\frac{235183}{20160}$	$-\frac{10754}{945}$	$\frac{135713}{20160}$	$-\frac{5603}{2520}$	$\frac{19087}{60480}$		$-\frac{5257}{17280}$
8	$\frac{16083}{4480}$	$-\frac{1152169}{120960}$	$\frac{242653}{13440}$	$-\frac{296053}{13440}$	$\frac{2102243}{120960}$	$-\frac{115747}{13440}$	$\frac{32863}{13440}$	$-\frac{5257}{17280}$	$\frac{1070017}{3628800}$

Diante dos conceitos expostos acima, podemos usar o seguinte código para resolver o problema de aproximar numericamente a solução da equação diferencial inicial usando o método de Adams-Bashforth, quando conhecemos os pontos inicialmente necessários.

```
h, f = self.h, self.f
for i in range(order, self.nsteps+1):
    if len(ans) == i:
        ans.append([0, 0])

    ans[i][1] = self.get_ab(ans, i, order)
    ans[i][0] = ans[i-1][0] + h

return ans
```

O método get_ab() utiliza os coeficientes adequados expostos na tabela acima para calcular o valor de Yn+1.

É importante notar que para k = 1, o método de Adams-Bashforth se torna o método de Euler simples.

3.2 Método de Adams-Multon

O método de Adams-Mutlon é muito similar ao apresentado acima, também é utilizado um polinômio de grau K para aproximar o a função $\frac{dy}{dt}$. A única diferença ocorre nos pontos selecionados para resolvermos o sistema de equações lineares encontrado. O último ponto escolhido, é o ponto (t_{n+1}, y_{n+1}) o que vai nos levar a uma equação final implícita para o problema de valor inicial proposto. Novamente, chegamos no impasse de termos em mãos uma equação implícita, que como já explicado, prejudica o desempenho do programo se for resolvida.

Usaremos então uma nova aproximação para o valor de y_{n+1}, que será obtida através do método de Adams-Bashforth de ordem K-1, onde K é o grau do polinômio usado no método de Adams-Multon. Segue abaixo a tabela de coeficientes utilizada para computar os valores necessários no código em python do projeto.

Coefficients and error constants for Adams–Moulton methods

k	β_0	β_1	β_2	β_3	β_4	β_5	β_6	β_7	C
0	1								$\frac{1}{2}$
1	$\frac{1}{2}$	$\frac{1}{2}$							$-\frac{1}{12}$
2	$\frac{5}{12}$	$\frac{2}{3}$	$-\frac{1}{12}$						$\frac{1}{24}$
3	$\frac{3}{8}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$					$-\frac{19}{720}$
4	$\frac{251}{720}$	$\frac{323}{360}$	$-\frac{11}{30}$	$\frac{53}{360}$	$-\frac{19}{720}$				$\frac{3}{160}$
5	$\frac{95}{288}$	$\frac{1427}{1440}$	$-\frac{133}{240}$	$\frac{241}{720}$	$-\frac{173}{1440}$	$\frac{3}{160}$			$-\frac{863}{60480}$
6	$\frac{19087}{60480}$	$\frac{2713}{2520}$	$-\frac{15487}{20160}$	$\frac{586}{945}$	$-\frac{6737}{20160}$	$\frac{263}{2520}$	$-\frac{863}{60480}$		$\frac{275}{24192}$
7	$\frac{5257}{17280}$	$\frac{139849}{120960}$	$-\frac{4511}{4480}$	$\frac{123133}{120960}$	$-\frac{88547}{120960}$	$\frac{1537}{4480}$	$-\frac{11351}{120960}$	$\frac{275}{24192}$	$-\frac{33953}{3628800}$

Aqui é também importante notar o fato que quando k = 1, temos o método já analisado anteriormente de Euler Inverso. A implementação é muito similar a apresentada no método de Adams-Bashforth.

3.3 Método da diferenciação inversa

Neste método, aproximamos diretamente a função y(t) que é solução para o problema de valor inicial apresentado, novamente por um polinômio de grau K. Usamos então os valores inicialmente fornecidos de Y para chegarmos a um sistema linear que nos permitirá descobrir os coeficiente do polinômio. Após todos terem sido determinados dessa forma, derivamos o polinômio encontrado e usamos a integral da equação 1.2 para novamente encontrar uma fórmula implícita para o valor de Yn+1. Diante dos mesmos problemas de equação implícitas, faremos uma predição do valor de Yn+1 pelo método de Adams-Bashforth de ordem K-1.

Para valores pequenos de K, segue a equação de Yn+1 em função dos pontos anteriores.

- BDF1: $y_{n+1} - y_n = hf(t_{n+1}, y_{n+1})$ (this is the backward Euler method)
- BDF2: $y_{n+2} - \frac{4}{3}y_{n+1} + \frac{1}{3}y_n = \frac{2}{3}hf(t_{n+2}, y_{n+2})$
- BDF3: $y_{n+3} - \frac{18}{11}y_{n+2} + \frac{9}{11}y_{n+1} - \frac{2}{11}y_n = \frac{6}{11}hf(t_{n+3}, y_{n+3})$
- BDF4: $y_{n+4} - \frac{48}{25}y_{n+3} + \frac{36}{25}y_{n+2} - \frac{16}{25}y_{n+1} + \frac{3}{25}y_n = \frac{12}{25}hf(t_{n+4}, y_{n+4})$
- BDF5: $y_{n+5} - \frac{300}{137}y_{n+4} + \frac{300}{137}y_{n+3} - \frac{200}{137}y_{n+2} + \frac{75}{137}y_{n+1} - \frac{12}{137}y_n = \frac{60}{137}hf(t_{n+5}, y_{n+5})$
- BDF6: $y_{n+6} - \frac{360}{147}y_{n+5} + \frac{450}{147}y_{n+4} - \frac{400}{147}y_{n+3} + \frac{225}{147}y_{n+2} - \frac{72}{147}y_{n+1} + \frac{10}{147}y_n = \frac{60}{147}hf(t_{n+6}, y_{n+6})$

Notamos aqui novamente que quando k = 1, temos o método de Euler Inverso. A implementação é bem similar ao método de Adams-Multon, adequando apenas os novos valores de coeficientes.

4. Comparação entre os métodos

4.0 Introdução

Nesta sessão, utilizaremos tabelas similares com a apresentada na sessão 1, que irão apresentar os valores calculados pelos métodos, e a diferença desses valores para a solução exata. Utilizaremos sempre a equação $\frac{dy}{dt} = 1 - t + 4y$.

4.1 Euler

Utilizando a expansão da série de Taylor, é possível provar que o erro local em um passo calculado pelo método de Euler é da ordem de h^3.

T	Y	Y_{exato} - Y
0.0	0.0	0.000000
0.1	0.1	-0.017217
0.2	0.23	-0.049789
0.3	0.402	-0.108022
0.4	0.6328	-0.208394
0.5	0.94592	-0.377028
0.6	1.37428	-0.655058
0.7	1.9640	-1.106868
0.8	2.77960	-1.832745
0.9	3.91144	-2.988223
1.0	5.486024	-4.813628

4.2 Euler Inverso

T	Y	Y_{exato} - Y
0.0	0.0	0.000000
0.1	0.130000	0.012783
0.2	0.318800	0.039011
0.3	0.599328	0.089306
0.4	1.02295	0.181756
0.5	1.669804	0.346856
0.6	2.664895	0.635549
0.7	4.20323	1.132359
0.8	6.58904	1.976691
0.9	10.29691	3.397241
1.0	16.06718	5.767527

4.3 Euler aprimorado

T	Y	Y_{exato} - Y
0.0	0.0000000000	0.000000
0.1	0.1150000000	-0.002217
0.2	0.2732000000	-0.006589
0.3	0.4953360000	-0.014686
0.4	0.8120972800	-0.029096
0.5	1.2689039744	-0.054044
0.6	1.9329778821	-0.096368
0.7	2.9038072655	-0.167064
0.8	4.3286347530	-0.283715
0.9	6.4253794344	-0.474290
1.0	9.5165615629	-0.783092

4.4 Runge-Kutta

T	Y	Y_{exato} - Y
0.0	0.0000000000	0.000000
0.1	0.1172000000	-0.000017
0.2	0.2797378133	-0.000051
0.3	0.5099075541	-0.000114
0.4	0.8409660953	-0.000227
0.5	1.3225238233	-0.000424
0.6	2.0285862046	-0.000759
0.7	3.0695496610	-0.001322
0.8	4.6100962143	-0.002253
0.9	6.8958875261	-0.003781
1.0	10.2933852856	-0.006268

4.5 Adams-Bashforth (K= 6, Utilizando Runge-Kutta para os pontos iniciais)

T	Y	Y_{exato} - Y
0.0	0.0000000000	0.000000
0.1	0.1172000000	-0.000017
0.2	0.2797378133	-0.000051
0.3	0.5099075541	-0.000114
0.4	0.8409660953	-0.000227
0.5	1.3225238233	-0.000424
0.6	2.0283613405	-0.000984
0.7	3.0687222083	-0.002149
0.8	4.6082802074	-0.004069
0.9	6.8923651509	-0.007304
1.0	10.2867228821	-0.012930

4.6 Adams-Muton (K= 6, Utilizando Runge-Kutta para os pontos iniciais)

T	Y	Y_{exato} - Y
0.0	0.0000000000	0.000000
0.1	0.1172000000	-0.000017
0.2	0.2797378133	-0.000051
0.3	0.5099075541	-0.000114
0.4	0.8409660953	-0.000227
0.5	1.3225247463	-0.000423
0.6	2.0285891209	-0.000756
0.7	3.0695561860	-0.001315
0.8	4.6101092212	-0.002240
0.9	6.8959118340	-0.003757
1.0	10.2934288499	-0.006224

4.6 Diferenciação inversa (K= 6, Utilizando Runge-Kutta para os pontos iniciais)

T	Y	Y_{exato} - Y
0.0	0.0000000000	0.000000
0.1	0.1172000000	-0.000017
0.2	0.2797378133	-0.000051
0.3	0.5099075541	-0.000114
0.4	0.8409660953	-0.000227
0.5	1.3226323758	-0.000316
0.6	2.0290249425	-0.000321
0.7	3.0706429776	-0.000228
0.8	4.6123514058	0.000002
0.9	6.9001657587	0.000497
1.0	10.3011413228	0.001488

Conclusão

Analisando as tabelas expostas na sessão anterior, vemos que os métodos de passo múltiplo possuem um erro bem inferior a maioria dos métodos de passo simples, como era de se esperar. Entretanto, como precisamos fazer mais operações, os métodos de passo múltiplos são mais lentos, além de acumular um erro maior de truncamento. Como vemos na tabela, o método que minimiza o valor absoluto do erro após 10 passos é o método da diferenciação inversa, com um erro de apenas 0,001. Se o desempenho for extremamente importante para o programa, poderíamos usar o método de Runge-Kutta, que é o método de passo simples que apresenta o menor valor absoluto de erro, no valor de 0,006. Diante disso podemos concluir que os métodos apresentados nesse texto conseguem aproximar bem a solução exata de uma equação diferencial ordinária.