Revista Eletrônica Paulista de Matemática

ISSN 2316-9664
Volume 12, jul. 2018

Solução numérica de equações diferenciais ordinárias usando Runge-Kutta: um estudo comparativo com Scilab

Numerical solution of ordinary differential equations using Runge-Kutta: a comparative study with Scilab

Francisco José Alves de Aquino
Instituto Federal do Ceará, Campus Fortaleza
fcoalves_aq@ifce.edu.br

Resumo

Neste artigo apresentamos a solução de diversas equações diferenciais de primeira ordem usando os métodos numéricos de Runge-Kutta de baixa, média e alta ordem, todos com passo fixo. Essas soluções numéricas são comparadas com as soluções analíticas, tendo como variação o passo de integração h e como critério de comparação o erro acumulado. Realizamos todos os cálculos e gráficos usando o *software* livre Scilab. De forma geral, os métodos de ordem mais alta são mais estáveis e apresentam melhor acurácia que os métodos de ordem mais baixa. Entretanto, se o passo h for muito pequeno, o erro total dos métodos pode ser essencialmente o mesmo. Este artigo pode servir como uma fonte de informação sobre os métodos Runge-Kutta para solução de equações diferenciais para outros pesquisadores ou professores da área de métodos numéricos.
Palavras-chave: Métodos numéricos. Runge-Kutta. Equações diferenciais. Simulação computacional. Scilab.

Abstract

In this paper we present the solution of three first order differential equations using low, medium and high order Runge-Kutta numerical method, all with fixed step size. These numerical solutions are compared with the analytical solutions, with the step of integration h and the cumulative error as a comparison criterion. We perform all calculations and graphs using the Scilab free software. In general, higher order methods are more stable and have better accuracy than lower order methods. However, if step h is too small, the total error of the methods can be essentially the same. This paper can serve as a source of information on Runge-Kutta methods for solving differential equations for other researchers or teachers in the field of numerical methods.
Keywords: Numerical methods. Runge-Kutta. Differential equations. Computational simulation. Scilab.

Artigo recebido em mar. 2018 e aceito em jun. 2018

1 Introdução

O modelamento de problemas de engenharia, economia, biologia, química ou física envolvem algum tipo de equação diferencial ou sistemas de equações diferenciais. Em muitas situações práticas, a solução analítica de uma equação diferencial ordinária (EDO) não é um problema trivial ou mesmo solucionável. Nestes casos, é necessário usarmos algum método numérico. Dentre os métodos numéricos disponíveis mais usados para solução de uma EDO estão os métodos Runge-Kutta (RK). Esses métodos são caracterizados por serem de passo único e por não ser necessário o cálculo das derivadas da função em estudo. O uso e estudo desses métodos começou com o famoso matemático Euler e foi intensificado no final do século XIX com o matemático C. Runge. A esses trabalhos pioneiros se seguiram as contribuições de K. Heun, W. Kutta e E. J. Nyström no início do século XX (BUTCHER, 1996). Com o advento dos computadores e *softwares* cada vez mais sofisticados, esse campo apresentou um crescimento extraordinário nas últimas décadas, sendo ainda uma área da matemática aplicada muito fértil.

Vejamos um exemplo tirado da engenharia elétrica: a relação entre corrente e tensão em um circuito elétrico composto por resistor, capacitor e indutor ligados em série alimentados por uma fonte de tensão independente gera uma EDO de ordem 2 com coeficientes constantes. Esse tipo de problema tem solução analítica e, em princípio, não necessitamos de método numérico para resolvê-lo. Já equação de um pêndulo oscilante, sem levarmos em conta a resistência do ar, é descrita por:

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\operatorname{sen}(\theta) = 0, \tag{1}$$

onde L é o comprimento do pêndulo, g é a constante gravitacional e θ é o ângulo formado pelo pêndulo com a vertical. Apesar de sua aparência simples, mesmo se conhecermos a posição e a velocidade em um dado instante, não poderemos resolver essa equação analiticamente. Para ângulos pequenos podemos usar a aproximação $\operatorname{sen}(\theta) \cong \theta$ e a equação 1 passa a ter uma solução analítica. Para os fundamentos básicos e uma introdução mais completa às equações diferenciais recomendamos a leitura de Boyce e Diprima (2002).

Neste artigo temos um duplo objetivo:

- apresentar alguns métodos RK de ordem elevada (> 4), que são pouco estudados nos cursos de bacharelado de engenharia e ciências exatas;
- comparar esses métodos RK de diferentes ordens em função da acurácia na solução de algumas equações diferenciais de teste.

Logo, usaremos EDOs cujas soluções analíticas sejam funções conhecidas. Como *software* de suporte à geração de gráficos, tabelas e solução numérica das equações faremos uso do *Scilab*®. O Scilab é um *software* livre que possui grande versatilidade para programação numérica, computação científica e geração de gráficos de forma muito semelhante ao *Matlab*®. A linguagem Scilab também tem alguma semelhança com a linguagem *C*, mas não é obrigatório ter uma área para declaração de variáveis, por exemplo (SCILAB, 2018).

Organizamos o restante deste artigo como segue. Na seção 2 mostramos como criar funções e gráficos usando o Scilab. A seção 3 traz os conceitos básicos sobre a solução numérica de EDOs e os métodos de Runge-Kutta implementados. Na seção 4 comentamos sobre erro entre a solução exata e a solução numérica, além de discutir sobre a estabilidade dos métodos de RK. Os resultados numéricos e uma breve discussão estão na seção 5. Na sequência temos as conclusões.

2 Scilab: criando funções e gerando gráficos

Frequentemente desejamos criar uma função nossa que será chamada no corpo do código de programa Scilab uma ou mais vezes. Para criar essa nova função no Scilab devemos usar a seguinte estrutura:

```
function [argumentos de saída] = nome-da-função(argumentos-de-entrada)
  declarações e cálculos
  declarações e cálculos
endfunction
```

Por exemplo (depois de "//" segue um comentário):

```
function [mmax, mmed, mmin, mgeo] = fxmi(dados) // esta função determina
  mmax = max(dados);      // os valores máximo,
  mmed = mean(dados);     // médio,
  mmin = min(dados);      // mínimo,
mgeo = sqrt(mmax*mmin); // e a média geométrica dos valores
endfunction               // máximo e mínimo do vetor de entrada.
```

O comando básico para gerar gráficos no Scilab é o "plot". Podemos incluir uma legenda com o comando "legend" (diversas curvas em um mesmo janela gráfica), título ("title") e os nomes dos eixos ("xlabel" e "ylabel"). Por exemplo:

```
close;  // fechando uma janela previamente aberta
x = 0:0.01:6;  // criando um vetor ''x'' de 0 a 6 com passo 0.01
y = 2*exp(-x/2).*sin(2*x); // função y
z = x.*(6-x).*cos(3*x)/4; // função z
w = 2*x./(x.*x + 1); // função w
plot(x,y,x,z,x,w); xgrid; // gráfico com o comando ''plot'' e mais uma malha de ''grid''.
legend('2*exp(-x/2).*sin(2*x)','x.*(6-x).*cos(3*x)/4','2*x./(x.*x + 1)'); // legenda
```

No endereço eletrônico https://help.scilab.org/docs/6.0.1/pt_BR/ temos um *help* que é bastante completo sobre o Scilab (SCILAB, 2018). Outras informações sobre o Scilab e o seu uso em métodos numéricos podemos encontrar, por exemplo, no livro colaborativo eletrônico organizado por pesquisadores da UFRS no *link* https://www.ufrgs.br/reamat/CalculoNumerico/livro-sci/main.html (JUSTO, 2018).

3 Solução numérica de uma EDO com RK

Uma EDO de primeira ordem tem a forma genérica

$$\frac{dy(t)}{dt} = f(t,y), \quad \text{para } t_0 \leq t \leq t_f, \tag{2}$$

obedecendo a uma condição inicial $y(t_0) = y_0$. Em geral, usaremos $t_0 = 0$. Sabemos que podemos aproximar o cálculo da derivada de uma função por

$$\frac{dy(t)}{dt} \cong \frac{y(t+dt) - y(t)}{dt} \tag{3}$$

Para solução numérica de uma EDO usamos um tempo discretizado em passos $h = dt = (t_f - t_0)/N$: $t_k = t_0 + kh$, sendo $k \geq 0$ um valor inteiro e N o número de pontos em que a função será calculada. Logo, a EDO 2 pode ser escrita como

$$y(t_{k+1}) \cong y(t_k) + hf(t_k, y_k), \text{ para cada } k = 0, 1, 2, \ldots, N. \tag{4}$$

O erro de truncamento local é da ordem de $h^2 : E_t = O(h^2)$. Esse é o conhecido método de Euler. Esse método de primeira ordem é pouco usado na prática, mas serve de inspiração para os demais métodos de métodos de Runge-Kutta de ordem mais alta.

3.1 Métodos Runge-Kutta: forma geral

Os métodos de Runge-Kutta têm o formato geral dado por

$$y(t_{k+1}) = y(t_k) + \sum_{n=1}^{N} c_n K_n \tag{5}$$

em que c_i são constantes ($\sum_{n=1}^{N} c_n = 1$) e os valores de K_i são calculados por

$$\begin{aligned}
K_1 &= hf(t_k, y_k) \\
K_2 &= hf(t_k + a_2 h, y_k + b_{21} K_1) \\
K_3 &= hf(t_k + a_3 h, y_k + b_{31} K_1 + b_{32} K_2) \\
&\vdots \\
K_N &= hf(t_k + a_N h, y_k + b_{N1} K_1 + b_{N2} K_2 + \cdots b_{NN-1} K_{N-1})
\end{aligned}$$

em que a_j e b_{ij} são constantes. Quanto maior o valor de N, maior a ordem do método e maior a sua acurácia (menor erro de truncamento). Podemos obter os valores dessas constantes igualando a equação 5 com a expansão da série de Taylor da equação diferencial. Esse processo leva a um sistema de equações não lineares como mais incógnitas que equações. Logo, podemos ter infinitas soluções para esses sistemas. Em geral, devemos escolher o valor de uma ou duas dessas incógnitas para determinar as demais. Para $N > 3$, a álgebra envolvida para resolver o sistema é muito laboriosa. Por exemplo, para $N = 3$ teremos um sistema com 6 equações e 8 incógnitas. Assim, precisamos estipular *a priori* o valor de duas dessas incógnitas e em seguida calcular as demais. Para um histórico bem detalhado sobre os métodos RK podemos consultar Butcher (1996).

3.2 Métodos Runge-Kutta de baixa ordem

Os métodos RK de ordem 2, 3 e 4 são bem conhecidos pelos estudantes de graduação e muitos livros textos de cálculo numérico apresentam esses métodos com algum detalhe como, por exemplo, Campos Filho (2007), Chapra e Canale (2008). Apresentamos duas versões para o RK2:

RK2 - Heun

$$K_1 = f(t_k, y_k) \quad (6)$$
$$K_2 = f(t_k + h, y_k + K_1 h) \quad (7)$$
$$y(t_{k+1}) = y(t_k) + \frac{h}{2}(K_1 + K_2) \quad (8)$$
$$t_{k+1} = t_k + h \quad (9)$$

RK2 - Ralston

$$K_1 = f(t_k, y_k) \quad (10)$$
$$K_2 = f(t_k + 3h/4, y_k + 3K_1 h/4) \quad (11)$$
$$y(t_{k+1}) = y(t_k) + \frac{h}{3}(K_1 + 2K_2) \quad (12)$$
$$t_{k+1} = t_k + h \quad (13)$$

Apresentamos também duas versões para o RK3:

RK3 - Heun

$$K_1 = f(t_k, y_k) \quad (14)$$
$$K_2 = f(t_k + h/3, y_k + K_1 h/3) \quad (15)$$
$$K_3 = f(t_k + 2h/3, y_k + 3K_2 h/3) \quad (16)$$
$$y(t_{k+1}) = y(t_k) + \frac{h}{4}(K_1 + 3K_3) \quad (17)$$
$$t_{k+1} = t_k + h \quad (18)$$

RK3 - Clássico

$$K_1 = f(t_k, y_k) \quad (19)$$
$$K_2 = f(t_k + h/2, y_k + K_1 h/2) \quad (20)$$
$$K_3 = f(t_k + h, y_k + (2K_2 - K_1)h) \quad (21)$$
$$y(t_{k+1}) = y(t_k) + \frac{h}{6}(K_1 + 4K_2 + K_3) \quad (22)$$
$$t_{k+1} = t_k + h \quad (23)$$

Já o método RK4 mais largamente usado é o método clássico (BURDEN e FAIRES, 2008):

$$K_1 = f(t_k, y_k) \quad (24)$$
$$K_2 = f(t_k + h/2, y_k + K_1 h/2) \quad (25)$$
$$K_3 = f(t_k + h/2, y_k + K_2 h/2) \quad (26)$$
$$K_4 = f(t_k + h, y_k + K_3 h) \quad (27)$$
$$y(t_{k+1}) = y(t_k) + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \quad (28)$$
$$t_{k+1} = t_k + h \quad (29)$$

Em geral, para problemas em que podemos usar um passo h pequeno, o método RK4 é satisfatório, pois o erro de truncamento local é da ordem de h^4. Entretanto, alguns problemas exigem muitas iterações ou um passo mais largo ou esses dois fatores ao mesmo tempo. Para essas situações, devemos levar em conta a necessidade de usarmos métodos de alta ordem. Para fins de simulação, implementamos os métodos de RK2-Ralston, RK3-Clássico e RK4-Clássico.

3.3 Métodos Runge-Kutta de alta ordem

Apresentaremos agora alguns métodos de ordem elevada. Podemos conferir a dedução desses métodos em diversos artigos: Butcher (1964), Luther (1968), Hairer (1978), Dormand e Prince (1980), Shampine (1986) e na tese de doutorado de Yaakub (1996). O pesquisador Butcher

(1964) recomenda como método de quinta ordem e seis etapas:

$$K_1 = f(t_k, y_k) \tag{30}$$
$$K_2 = f(t_k + h/4, y_k + K_1 h/4) \tag{31}$$
$$K_3 = f(t_k + h/4, y_k + (K_1 + K_2)h/8) \tag{32}$$
$$K_4 = f(t_k + h/2, y_k + (2K_3 - K_2)h/2) \tag{33}$$
$$K_5 = f(t_k + 3h/4, y_k + (9K_4 + 3K_1)h/16) \tag{34}$$
$$K_6 = f(t_k + h, y_k + (-3K_1 + 2K_2 + 12K_3 - 12K_4 + 8K_5)h/7) \tag{35}$$
$$y(t_{k+1}) = y(t_k) + \frac{h}{90}(7K_1 + 32K_3 + 12K_4 + 32K_5 + 7K_6) \tag{36}$$
$$t_{k+1} = t_k + h \tag{37}$$

Podemos equacionar o método RK-Nyström, também de quinta ordem e seis etapas, como segue (YAAKUB, 1996):

$$K_1 = f(t_k, y_k) \tag{38}$$
$$K_2 = f(t_k + h/3, y_k + K_1 h/3) \tag{39}$$
$$K_3 = f(t_k + 2h/5, y_k + (4K_1 + 6K_2)h/25) \tag{40}$$
$$K_4 = f(t_k + h, y_k + (k_1 - 12K_2 + 15K_3)h/4) \tag{41}$$
$$K_5 = f(t_k + 2h/3, y_k + (6K_1 + 90K_2 - 50K_3 + 8K_4)h/81) \tag{42}$$
$$K_6 = f(t_k + 4h/5, y_k + (6K_1 + 36K_2 + 10K_3 - 8K_4)h/75) \tag{43}$$
$$y(t_{k+1}) = y(t_k) + \frac{h}{192}(23K_1 + 125K_3 - 81K_5 + 125K_6) \tag{44}$$
$$t_{k+1} = t_k + h \tag{45}$$

Um outro método RK de quinta ordem foi proposto por Dormand e Prince (1980):

$$K_1 = f(t_k, y_k) \tag{46}$$
$$K_2 = f(t_k + h/5, y_k + K_1 h/5) \tag{47}$$
$$K_3 = f(t_k + 3h/10, y_k + (3K_1 + 9K_2)h/40) \tag{48}$$
$$K_4 = f(t_k + 4h/5, y_k + (44k_1 - 168K_2 + 160K_3)h/45) \tag{49}$$
$$K_5 = f(t_k + 8h/9, y_k + (19372K_1 - 76080K_2 + 64448K_3 - 1908K_4)h/6561) \tag{50}$$
$$K_6 = f\left(t_k + h, y_k + (\frac{9017K_1}{3168} - \frac{355K_2}{33} + \frac{46732K_3}{5247} + \frac{49K_4}{176} - \frac{5101K_5}{18656})h\right) \tag{51}$$
$$y(t_{k+1}) = y(t_k) + h(35K_1/384 + 500K_3/1113 + 125K_5/192 - 2187K_6/6784 + 11K_6/84) \tag{52}$$
$$t_{k+1} = t_k + h \tag{53}$$

Já o método proposto por Anton Huta de sexta ordem (com 8 etapas) tem as seguintes expressões

(YAAKUB, 1996):

$$K_1 = f(t_k, y_k) \tag{54}$$
$$K_2 = f(t_k + h/9, y_k + K_1 h/9) \tag{55}$$
$$K_3 = f(t_k + h/6, y_k + (K_1 + 3K_2)h/24) \tag{56}$$
$$K_4 = f(t_k + h/3, y_k + (k_1 - 3K_2 + 4K_3)h/6) \tag{57}$$
$$K_5 = f(t_k + h/2, y_k + (-5K_1 + 27K_2 - 24K_3 + 6K_4)h/8) \tag{58}$$
$$K_6 = f(t_k + 2h/3, y_k + (221K_1 - 981K_2 + 867K_3 - 102K_4 + K_5)h/9) \tag{59}$$
$$K_7 = f(t_k + 5h/6, y_k + (-183K_1 + 678K_2 - 472K_3 - 66K_4 + 80K_5 + 3K_6)h/48) \tag{60}$$
$$K_8 = f(t_k + h, y_k + (716K_1 - 2079K_2 + 1002K_3 + 834K_4 - 454K_5 - 9K_6 + 72K_7)h/82) \tag{61}$$
$$y(t_{k+1}) = y(t_k) + \frac{h}{840}(41K_1 + 216K_3 + 27K_4 + 272K_5 + 27K_6 + 216K_7 + 41K_8) \tag{62}$$
$$t_{k+1} = t_k + h \tag{63}$$

No endereço eletrônico http://sce.uhcl.edu/rungekutta/ podemos encontrar os coeficientes de métodos de ordem ainda mais alta (10, 12, 14). No método de décima ordem precisamos de 17 etapas (K_1 a K_{17}). Implementamos o método RK10 com o seguinte trecho de código Scilab (coeficientes no *link* http://sce.uhcl.edu/rungekutta/rk108.txt) (UNIVERSITY OF HOUSTON, c2016):

```
vk = 0*vk;  // vetor com coeficientes ''Ki''
x = xk;  // instante de tempo ''t''
for kk=1:17  // calculando as etapas
    y = y10;  // variável auxiliar que guarda ''y(tk)''
    x = xk + a(kk)*h;  // passo no tempo
    for kb=1:kk  // calculando ''y'' nas diversas etapas
       y = y + vk(kb)*b(kk,kb);
    end;
    vk(kk)=h*fd(x,y);  // cálculo de ''Ki'' na etapa ''i''
end;
y10 = y10 + sum(c.*vk);  // novo valor de ''y(tk)'', c: vetor com pesos dos ''Ki''
xk = xk + h;  // novo instante de tempo - próxima iteração.
```

O código acima não está otimizado. São mais 100 coeficientes (“a(kk)”, “b(kk,kb)” e “c”) diferentes de zero para usarmos o RK10.

4 Erro e estabilidade

De forma simplista, podemos afirmar que quanto menor o passo h maior será o esforço computacional necessário para resolver a EDO no intervalo definido (t_0, t_f) e menor será o erro acumulado. Entretanto, se o passo h for demasiado pequeno, os erros de arredondamento serão acumulados e o erro total será significativo. Então, sem levarmos em conta o esforço computacional, podemos estimar um valor ótimo para o passo h de forma que o erro total (truncamento mais arredondamento) seja mínimo.

Indo na outra direção, se o passo h for muito grande, além do erro maior devido à aproximação grosseira da função, podemos chegar em uma região de instabilidade e o resultado divergir completamente do valor esperado. Consideremos o problema básico

$$\begin{aligned}\frac{dy}{dt} = \lambda y \rightarrow y_{k+1} &= (1+h\lambda)y_k \\ y_{k+1} &= (1+z)y_k \\ y_{k+1} &= Q(z)y_k\end{aligned}$$

com $z = h\lambda$. Podemos calcular a região de estabilidade dos métodos RK explícitos de passo único usando a desigualdade $|Q(z)| < 1$, sendo $Q(z)$ expresso por:

$$Q(z) = 1+z \qquad \text{RK1 - método de Euler} \tag{64}$$

$$Q(z) = 1+z+\frac{z^2}{2!} \qquad \text{RK2} \tag{65}$$

$$Q(z) = 1+z+\frac{z^2}{2!}+\frac{z^3}{3!} \qquad \text{RK3} \tag{66}$$

$$Q(z) = 1+z+\frac{z^2}{2!}+\frac{z^3}{3!}+\frac{z^4}{4!} \qquad \text{RK4} \tag{67}$$

De forma geral, o polinômio $Q(z)$ para o cálculo da região de convergência de um método RK explícito de ordem N com R ($R \geq N$) etapas é expresso por:

$$Q(z) = 1+z+\frac{z^2}{2!}+\frac{z^3}{3!}+\cdots\frac{z^N}{N!} \tag{68}$$

Para os métodos de RK de ordem 1 a 4, podemos conferir a região de convergência na Figura 1. Notamos que a região de estabilidade cresce com a ordem do método de RK.

5 Resultados numéricos e discussão

Usaremos as funções indicadas na Tabela 1 e diversos passos h para cada um dos métodos RK que descrevemos acima. Como principal critério de comparação adotaremos o erro acumulado

$$e_a = \sum_k |y_{KN}(t_k) - y_a(t_k)|, \tag{69}$$

da solução numérica $y_{KN}(t_k)$ em relação à resposta analítica $y_a(t)$. Para um melhor uso do espaço neste artigo, apresentamos os resultados na forma de gráficos e não na forma de tabelas comparativas.

Tabela 1: Funções de teste

Equações diferenciais		Soluções analíticas
$\frac{dy}{dt} = 1+5(t-y)$	$\Rightarrow$	$y(t) = t - e^{-5t}$
$\frac{dy}{dt} = \frac{2t}{y} - ty$	$\Rightarrow$	$y(t) = \sqrt{2-e^{-x^2}}$
$\frac{dy}{dt} = 10 - 0,1y^2$	$\Rightarrow$	$y(t) = 10\frac{1-e^{-2t}}{1+e^{-2t}}$

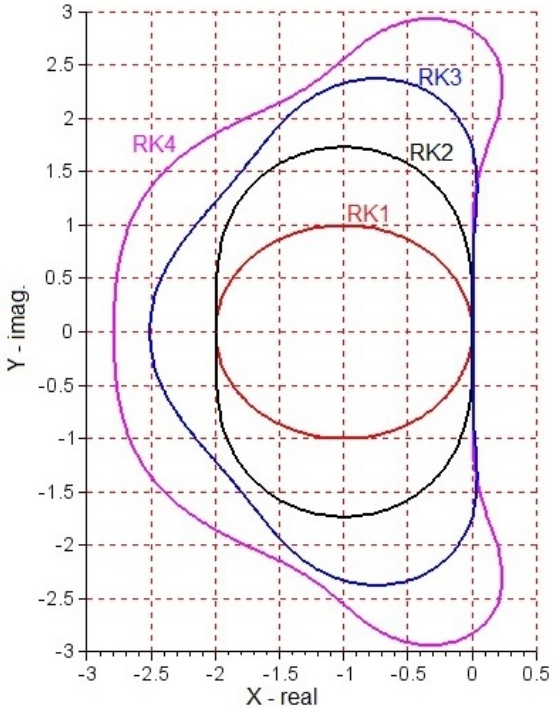

Figura 1: Regiões de convergência.

Apresentamos na Figura 2 a solução da EDO $y' = 1+5(t-y)$ com passo relativamente grande ($h = 0,40$). Observamos que o método RK2 não apresenta uma boa solução. O método RK3 tem um erro inicial grande, mas diminui com o aumento das iterações. Na Figura 3 temos os desempenhos (erro absoluto) dos métodos em relação à EDO $y' = 1+5(t-y)$ para um passo $h = 0,20$. Podemos perceber que mesmo quando o passo é relativamente grande, isto é $h = 0,20$, as diferenças entre os valores teóricos dados pela solução analítica e os valores calculados pelos métodos de RK são menores que 0,15. Quando $t \to +\infty$, todos os métodos convergem para o valor analítico. De forma geral, quanto maior a ordem do método RK que empregamos, menor o erro acumulado. Entretanto, para essa EDO, o desempenho do método RK-6 não é significativamente superior aos métodos de quinta ordem. Como esperado, o método RK-10 tem um desempenho muito superior, desde que o passo h seja maior que $1,25 \times 10^{-2}$. Para um passo muito pequeno, os erros de arredondamento começam a sobrepujar os erros de truncamento. Cada método apresenta um valor ótimo para o passo h tal que o erro total seja mínimo. Podemos extrair todas essas informações da Figura 4.

Já para EDO $y' = \frac{2t}{y} - ty$, com o passo $h = 0,20$, o método RK2 diverge para $t \to +\infty$. O método RK10 também apresenta um desempenho relativamente ruim com esse passo grande, somente para $h \leq 0,2$ sua acurácia é melhor que de todos os outros métodos. Para um passo h menor que $6,25 \times 10^{-3}$ o desempenho do RK10 começa a piorar. Os métodos de quinta ordem apresentam um desempenho similar, o método de sexta ordem é claramente superior aos de ordem mais baixa. As figuras 5 e 6 mostram o desempenho de cada um dos métodos.

Para EDO $y' = 10 - 0,1y^2$, mesmo um passo $h = 0,40$ não afeta a convergência de nenhum dos métodos, todos conseguem uma boa aproximação com a solução analítica. As figuras 7 e 8 mostram o desempenho dos métodos para outros valores do passo h para esta EDO. Podemos

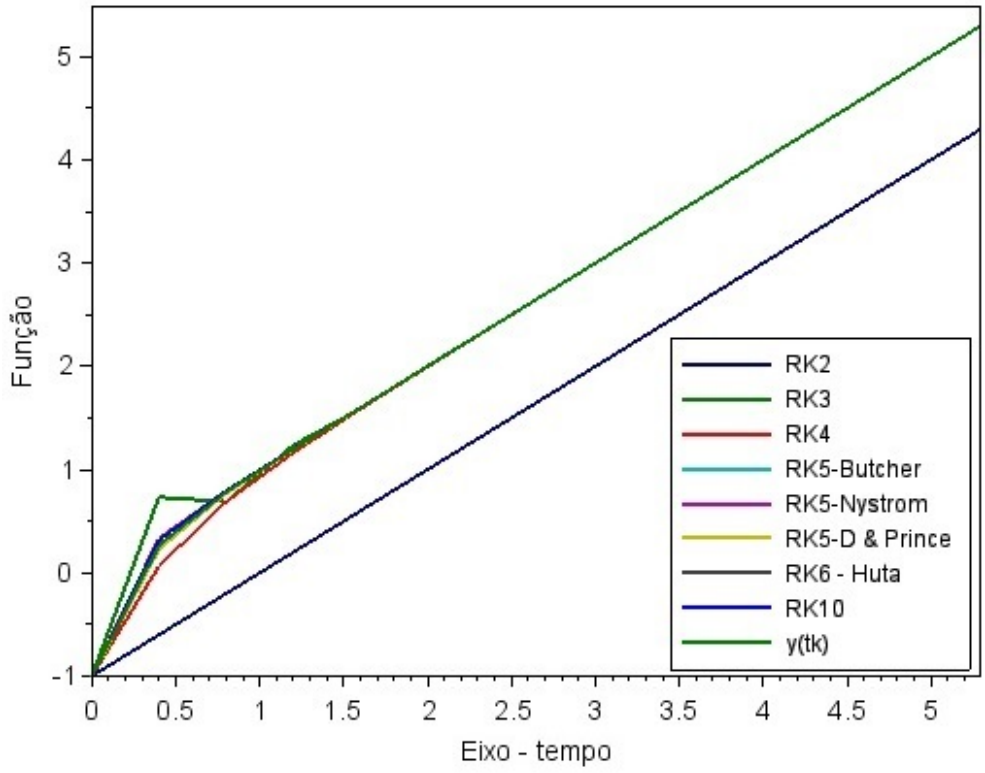

Figura 2: Solução em relação à EDO $y' = 1 + 5(t - y)$ para $h = 0,40$.

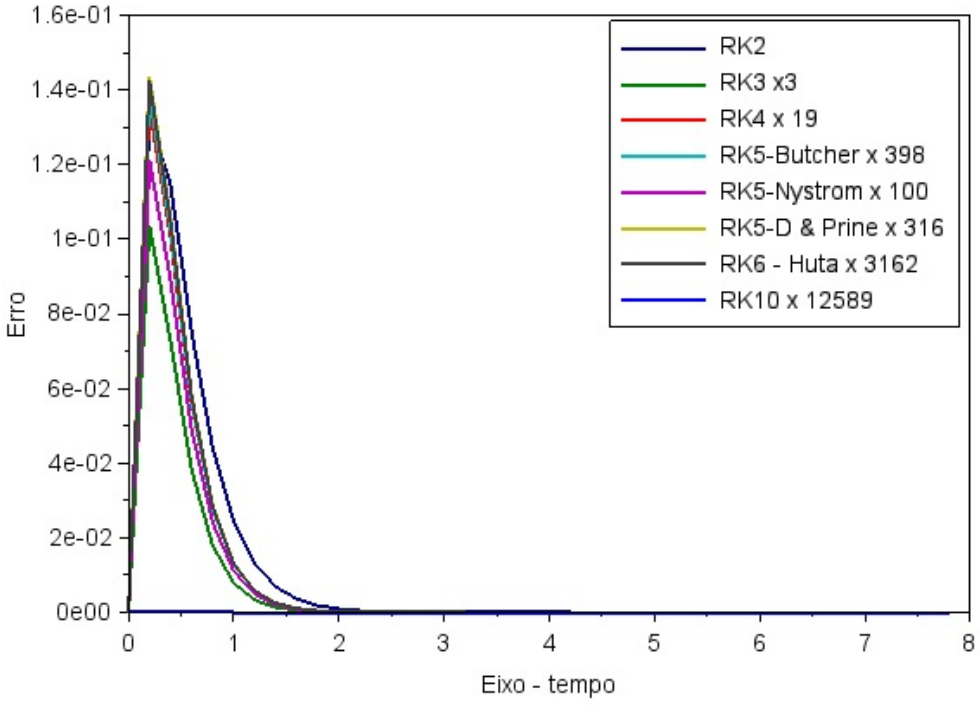

Figura 3: Desempenho em relação à EDO $y' = 1 + 5(t - y)$. Erro absoluto para $h = 0,20$.

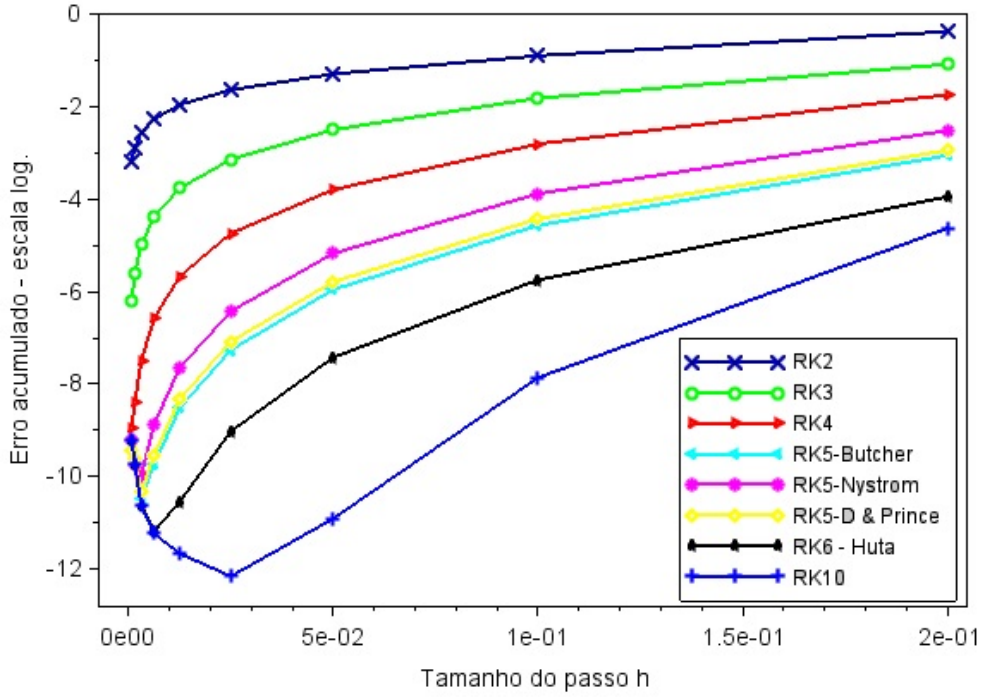

Figura 4: Desempenho em relação à EDO $y' = 1 + 5(t - y)$. Erro acumulado.

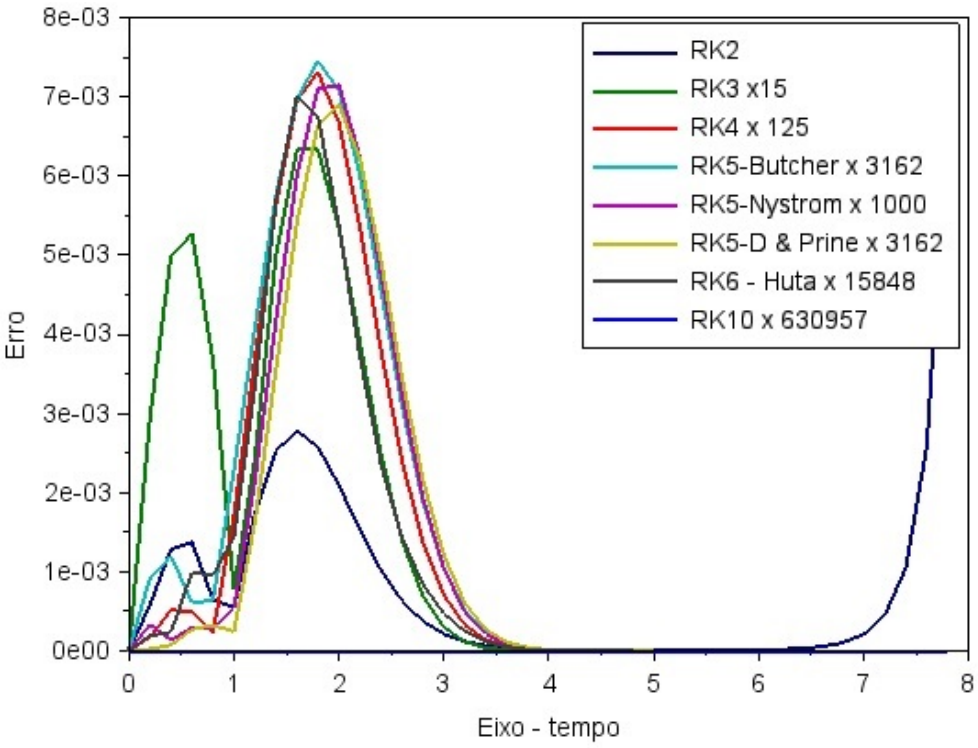

Figura 5: Desempenho em relação à EDO $y' = \frac{2t}{y} - ty$. Erro absoluto para $h = 0,20$.

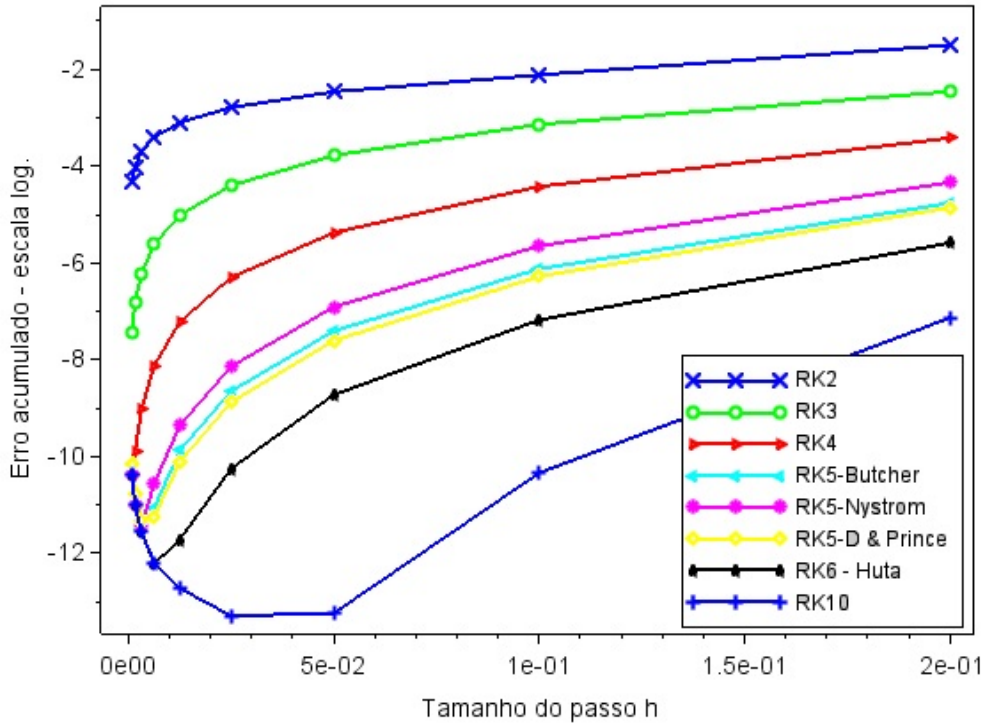

Figura 6: Desempenho em relação à EDO $y' = \frac{2t}{y} - ty$. Erro acumulado.

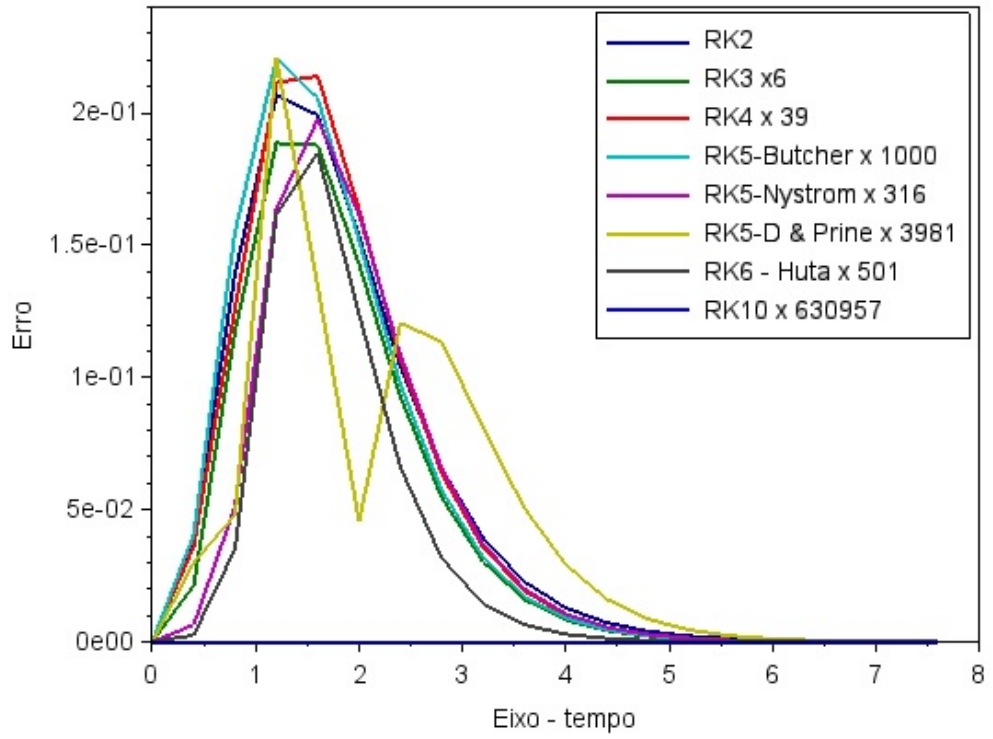

Figura 7: Desempenho em relação à EDO $y' = 10 - 0,1y^2$. Erro absoluto para $h = 0,40$.

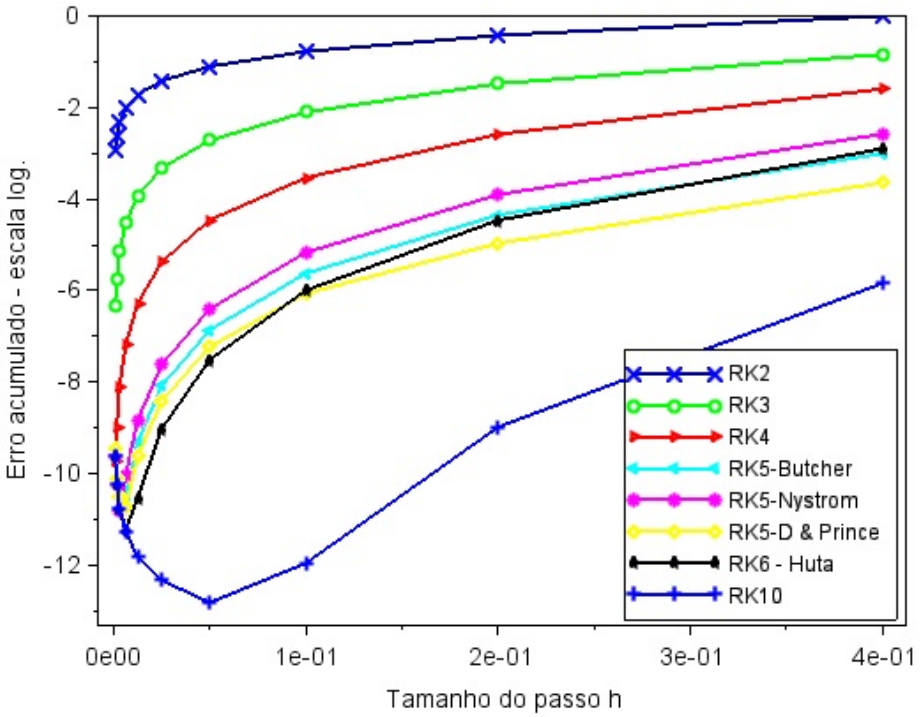

Figura 8: Desempenho em relação à EDO $y' = 10 - 0,1y^2$. Erro acumulado.

notar que para esta EDO o método de RK6 de Huta não apresentou um desempenho melhor que o métodos de quinta ordem de Butcher e Dormand-Prince para $h > 0,05$. Como esperado, o método de décima ordem superou todos os outros para um passo $h \geq 6,25 \times 10^{-3}$.

6 Conclusão

Neste artigo apresentamos e comparamos diversos métodos de Runge-Kutta para a solução de algumas equações difereciais de primeira ordem. Alguns métodos já são bem conhecidos dos estudantes da graduação, mas os métodos de ordem mais elevada encontramos somente na literatura mais especializada. Implementamos todos os métodos usando o Scilab que é um *software* livre. Da análise dos resultados podemos concluir que, de forma geral, um método de ordem mais alta apresenta um desempenho melhor que um método de ordem mais baixa, desde que o passo h não seja muito pequeno. Se o passo for muito pequeno, então o acumulo de erros de arredondamento acabam por aumentar o erro total. Assim, existe um passo ótimo para cada método e EDO. Não existe uma diferença muito significativa entre o método de quinta ordem proposto por Dormand-Prince e o de sexta ordem (método de Huta) que implementamos; para métodos de ordem elevada (maior ou igual a 10), a quantidade de memória que usamos para armazenar os coeficientes e o número de vezes que avaliamos a função $f(t,y)$ é muito maior que nos métodos de baixa ordem (2, 3 e 4). A decisão final de qual método RK usar cabe ao pesquisador após analisar o problema que está resolvendo, mas, de forma geral, um método quarta ou quinta ordem, com um passo h adequado, deve ter acurácia suficiente. Somente quando a quantidade de iterações for muito elevada é que se torna imperativo o uso de um método RK de décima ordem ou ainda maior.

7 Referências bibliográficas

BOYCE, W. E.; DIPRIMA, R. C. **Equações diferenciais elementares e problemas de valores de contorno**. 7. ed. Rio de Janeiro, RJ: LTC: 2002.

BURDEN, R. L.; FAIRES, J. D. **Análise numérica**. São Paulo: Cengage Learning, 2008.

BUTCHER, J. C. On Runge-Kutta processes of high order. **Journal of the Australian Mathematical Society**, v. 4, n. 2, p. 179-194, 1964. https://doi.org/10.1017/S1446788700023387.

BUTCHER, J. C. A history of Runge-Kutta methods. **Applied Numerical Mathematics**, v. 20, n. 3, p. 247-260, 1996. https://doi.org/10.1016/0168-9274(95)00108-5.

CAMPOS FILHO, F. F. **Algoritmos numéricos**. 2. ed. Rio de Janeiro: LTC, 2007.

CHAPRA, S. C.; CANALE, R. P. **Métodos numéricos para engenharia**. 5. ed. São Paulo: McGraw-Hill, 2008.

DORMAND, J. R.; PRINCE, P. J. A family of embedded Runge-Kutta formulae. **Journal of Computational and Applied Mathematics**, v. 6, n. 1, p. 19-26, 1980. https://doi.org/10.1016/0771-050X(80)90013-3.

HAIRER, E. A Runge-Kutta Method of Order 10, **IMA Journal of Applied Mathematics**, v. 21, n. 1, p. 47-59, 1978. https://doi.org/10.1093/imamat/21.1.47.

JUSTO, D. A. R. et al. (Org.). **Cálculo numérico**: um livro colaborativo: versão Scilab. 2018. Disponível em: https://www.ufrgs.br/reamat/CalculoNumerico/livro-sci/main.html. Acesso em: 10 fev. 2018.

LUTHER, H. A. An explicit sixth-order Runge-Kutta formula. **Mathematics of Computation**, v. 22, p. 434-436, 1968. https://doi.org/10.1090/S0025-5718-68-99876-1.

SCILAB. Open source software for numerical computation. Disponível em: http://www.scilab.org. Acesso em: 19 fev. 2018.

SHAMPINE, L. F. Some practical Runge-Kutta formulas. **Mathematics of Computation**, v. 46, n. 173, p. 135-150, 1986. Disponível em: http://www.ams.org/journals/mcom/1986-46-173/S0025-5718-1986-0815836-3/. Acesso em: 19 fev. 2018.

University of Houston. **College of Science and Engeneering**. Houston, c2016. Disponível em: https://www.uhcl.edu/scienceengineering/. Acesso em: 19 fev. 2018.

YAAKUB, A. R. Computer solution of non-linear integration formula for solving initial value problems. 1996. 382 f. Tese (Doutorado em Filosofia) - Loughborough University, Departamento de Estudos da Computação, Loughborough, 1996.