Capítulo 4

Solução Numérica de Equações Diferenciais Ordinárias

Uma equação diferencial ordinária tem a forma geral :

$$F(x, y, \frac{dy}{dx}, \frac{d^2y}{dx^2}, \frac{d^3y}{dx^3}, \ldots, \frac{d^ny}{dx^n}) = 0$$

A equação apresentada é chamada de equação diferencial ordinária de n-ésima ordem. Ela é uma equação ordinária porque há somente uma variável independente, x. É de n-ésima ordem porque a maior derivada é de ordem n.

Uma função $y(x)$, n vêzes diferenciável, satisfazendo a equação anterior é chamada de solução desta equação. As equações diferenciais ordinárias têm várias soluções. É necessário que sejam dadas informações adicionais sobre $y(x)$ e/ou sobre suas derivadas em valores específicos de x para que ela seja a solução única. Para uma equação diferencial de ordem n, normalmente são suficientes n condições adicionais para garantir que a solução $y(x)$ seja única. Se todas as n condições adicionais forem especificadas para um mesmo valor de x, x_0 por exemplo, temos um problema conhecido como Problema do Valor Inicial, **PVI**. Caso estas n condições adicionais sejam dadas para mais de um valor de x, temos um problema conhecido como Problema de Valor de Contorno, **PVC**.

Em geral, é difícil a obtenção de soluções analíticas para equações diferenciais. Na maioria dos casos as soluções devem ser geradas através de métodos numéricos. Neste capítulo, apresentaremos diversos métodos para resolução numérica de equações diferenciais ordinárias. Inicialmente, iremos tratar apenas com equações diferenciais de 1ª ordem. os métodos apresentados serão estendidos depois para tratar com equações diferenciais de ordem superior.

4.1 Método de Euler

Vamos considerar equações diferencias escritas na forma :

$$F(x, y, \frac{dy}{dx}) = 0$$

Esta equação pode ser escrita como :

$$\frac{dy}{dx} = f(x, y)$$

ou

$$y' = f(x, y)$$

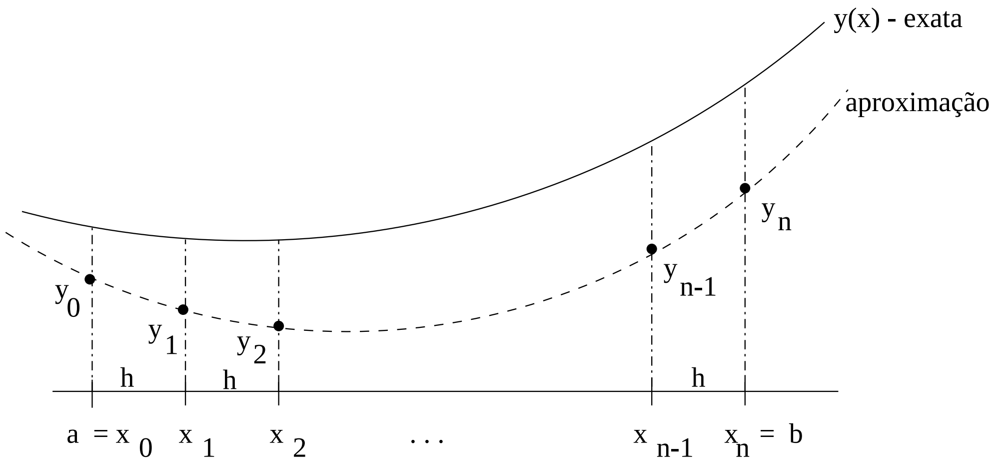
Como vimos, o objetivo da resolução de equações diferenciais ordinárias é a obtenção de uma solução, $y(x)$, dada uma condição inicial específica, considerando a variável independente em um intervalo $[a, b]$. O intervalo de integração $[a, b]$, é dividido em sub-intervalos e o valor exato da solução é aproximado em $n + 1$ valores espaçados de x,

$$x_k = x_0 + kh \qquad k = 0, 1, 2, \ldots, n$$

com

$$h = \frac{b - a}{n}$$

sendo h chamado de passo de integração. A figura mostra os conceitos apresentados.



O método de Euler consiste na geração de aproximações sucessivas para $y(x)$, utilizando a expressão :

$$y_{k+1} = y_k + hf(x_k, y_k) \qquad k = 0, 1, 2, \ldots, n$$

O método de Euler pode ser implementado através do algoritmo :

Algorithm 1: Método de Euler

Entrada $[a, b]$, h e $y0$
Fazer $x0 = a$
Fazer $n = (b - a)/h$
for k = 0 to n **do**
  Calcular $y_{k+1} = y_k + hf(x_k, y_k)$
  Calcular $x_{k+1} = x0 + kh$
end for
Apresentar valores de x_k e y_k

Utilizando o Scilab, vamos aplicar o método de Euler para resolver a equação :

$$\frac{dy}{dx} = y$$

com condição inicial $y(0) = 1$, no intervalo $[0, 1]$. A solução exata desta equação é :

$$y(x) = e^x$$

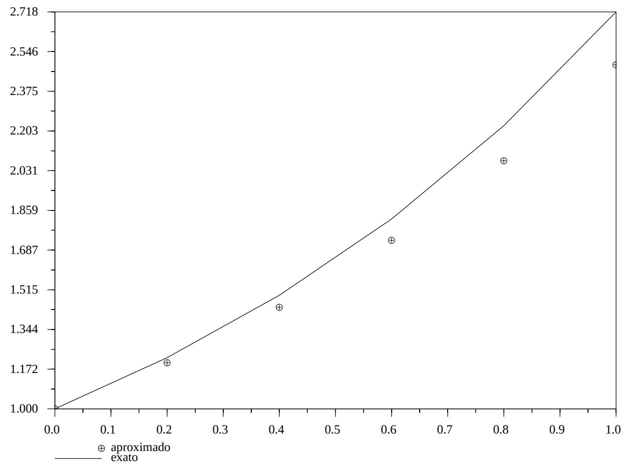
Considerando $h = 0.2$, temos :

```
-->getf('f1.sci')      // arquivo com a funcao f(x,y) = y

-->getf('euler.sci')   // metodo de euler

-->// Solucao para [a,b] = [0,1], h = 0.2 e y(0) = 1

-->euler(0, 1, 0.2, 1)
 ans  =

!   0.     1.         1.        !
!   0.2    1.2        1.2214028 !
!   0.4    1.44       1.4918247 !
!   0.6    1.728      1.8221188 !
!   0.8    2.0736     2.2255409 !
!   1.     2.48832    2.7182818 !
```

Na primeira coluna são mostrados os valores de x, na segunda coluna são mostrados os valores de y e na terceira coluna são mostrados os valores de $exp(x)$.

A curva exata e os valores aproximados obtidos são mostradas no gráfico[1]:

[1] gerado através do comando **plot2d([x;x]',[y;exp(x)]',[-3,1],"121",'@aproximado@exato')**



4.2 Método de Heun

O método de Heun consiste na geração de aproximações para a solução da equação diferencial

$$\frac{dy}{dx} = f(x, y)$$

com $y(x_0) = y_0$ no intervalo $[a, b]$ utilizando as equações :

$$p_{k+1} = y_k + hf(x_k, y_k)$$

e

$$\begin{aligned} y_{k+1} &= y_k + \frac{h}{2}[f(x_k, y_k) + f(x_{k+1}, p_{k+1}) \\ &= y_k + \frac{h}{2}[f(x_k, y_k) + f(x_{k+1}, y_k + hf(x_k, y_k))] \end{aligned}$$

O método de Heun pode ser implementado utilizando o seguinte algoritmo :

Algorithm 2: Método de Heun

Entrada $[a, b]$, h e $y0$
Fazer $x0 = a$
Fazer $n = (b - a)/h$
for k = 0 to n **do**
  Calcular $x_{k+1} = x0 + kh$
  Calcular $y_{k+1} = y_k + \frac{h}{2}[f(x_k, y_k) + f(x_{k+1}, y_k + hf(x_k, y_k))]$
end for
Apresentar valores de x_k e y_k

Vamos resolver a equação do exemplo anterior :

$$\frac{dy}{dx} = y$$

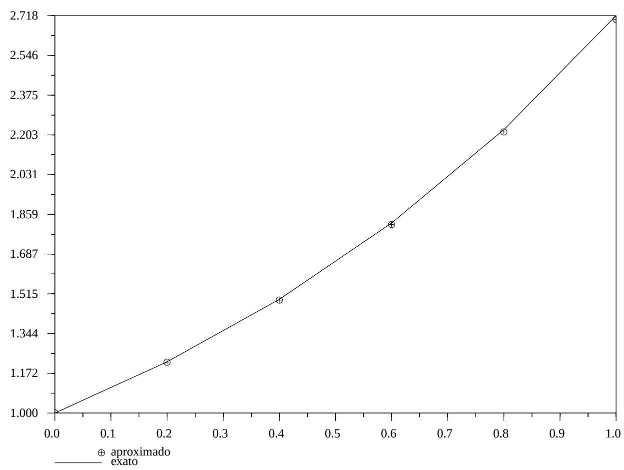
com condição inicial $y(0) = 1$, no intervalo $[0, 1]$, utilizando o método de Heun.

Os resultados obtidos no Scilab são :

```
-->getf('f1.sci')     // arquivo com a funcao f(x,y) = y

-->getf('heun.sci')   // metodo de heun

-->heun(0,1,0.2,1)
 ans  =

!   0.     1.           1.        !
!   0.2    1.22         1.2214028 !
!   0.4    1.4884       1.4918247 !
!   0.6    1.815848     1.8221188 !
!   0.8    2.2153346    2.2255409 !
!   1.     2.7027082    2.7182818 !

-->
```

Na primeira coluna são mostrados os valores de x e na segunda coluna são mostrados os valores de y.

A curva exata e os valores aproximados obtidos são mostradas no gráfico[2]:



Podemos observar que o método de Heun produz resultados melhores que o método de Euler.

[2]gerado através do comando **plot2d([x;x]',[y;exp(x)]',[-3,1],"121",'@aproximado@exato')**

4.3 Métodos de Runge-Kutta

Os métodos de Runge-Kutta utilizam valores de $f(x, y)$ no intervalo $[x_n, x_n + h]$ para obter uma boa aproximação para y_{n+1}. O método de Heun é também conhecido como método de Runge-Kutta de 2ª ordem. Outras equações utilizadas são :

1. Runge-Kutta de 3ª ordem :

$$y_{k+1} = y_k + \frac{h}{6}[f_1 + 4f_2 + f_3]$$

   com os coeficiente f dados por :

$$\begin{aligned} f_1 &= f(x_k, y_k) \\ f_2 &= f(x_k + \frac{h}{2}, y_k + \frac{h}{2} f_1) \\ f_3 &= f(x_k + h, y_k + 2hf_2 - hf_1) \end{aligned}$$

2. Runge-Kutta de 4ª ordem :

$$y_{k+1} = y_k + \frac{h}{6}(f_1 + 2f_2 + 2f_3 + f_4)$$

   com os coeficientes f dados por :

$$\begin{aligned} f_1 &= f(x_k, y_k) \\ f_2 &= f(x_k + \frac{h}{2}, y_k + \frac{h}{2} f_1) \\ f_3 &= f(x_k + \frac{h}{2}, y_k + \frac{h}{2} f_2) \\ f_4 &= f(x_k + h, y_k + hf_3) \end{aligned}$$

O método de 4ª ordem é o mais utilizado. Pode ser implementado utilizando o seguinte algoritmo :

Algorithm 3: Método de Runge-Kutta de 4ª ordem

Entrada $[a, b]$, h e $y0$
Fazer $x0 = a$
Fazer $n = (b - a)/h$
for k = 0 to n **do**
  Calcular f_1, f_2, f_3 e f_4
  Calcular $y_{k+1} = y_k + \frac{h}{6}(f_1 + 2f_2 + 2f_3 + f_4)$
  Fazer $x_{k+1} = x_k + h$
end for
Apresentar valores de x_k e y_k

Vamos considerar a equação diferencial

$$\frac{dy}{dx} = (x - y)/2$$

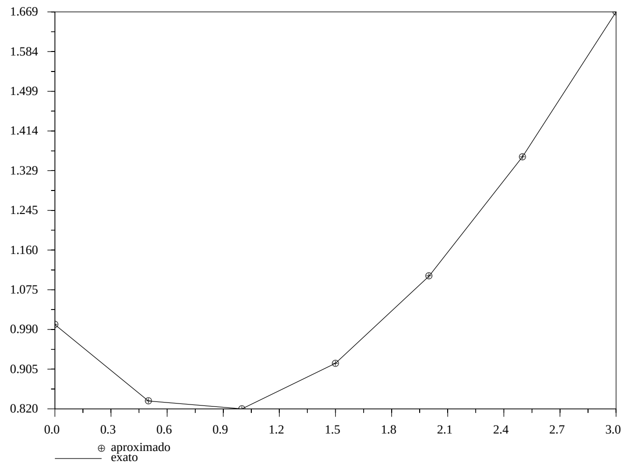
com condição inicial $y(0) = 1$. A solução exata é $y(x) = 3e^{-x/2} + x - 2$. Utilizando o método de Runge-Kutta de $4^{\underline{a}}$ ordem, obtemos a solução aproximada para $y(x)$ no intervalo $[0, 3]$ para $h = 1/85$. Os resultados apresentados usando o Scilab são :

```
-->getf('f2.sci')        // arquivo com a funcao f(x,y) = (x - y)/2

-->getf('rk4.sci')       // metodo de Runge-Kutta 4a. ordem

-->rk4(0, 3, 1/8, 1)
 ans  =

!   0.        1.           1.         !
!   0.125     0.9432392    0.9432392 !
!   0.25      0.8974908    0.8974907 !
!   0.375     0.8620874    0.8620874 !
!   0.5       0.8364024    0.8364023 !
!   0.625     0.8198470    0.8198469 !
!   0.75      0.8118679    0.8118678 !
!   0.875     0.8119457    0.8119456 !
!   1.        0.8195921    0.8195920 !
!   1.125     0.8343486    0.8343485 !
!   1.25      0.8557844    0.8557843 !
!   1.375     0.8834949    0.8834947 !
!   1.5       0.9170998    0.9170997 !
!   1.625     0.9562421    0.9562419 !
!   1.75      1.0005862    1.0005861 !
!   1.875     1.049817     1.0498169 !
!   2.        1.1036385    1.1036383 !
!   2.125     1.1617724    1.1617723 !
!   2.25      1.2239575    1.2239574 !
!   2.375     1.2899485    1.2899483 !
!   2.5       1.3595145    1.3595144 !
!   2.625     1.4324392    1.432439   !
!   2.75      1.5085189    1.5085188 !
!   2.875     1.5875626    1.5875625 !
!   3.        1.6693906    1.6693905 !

-->
```

Na primeira coluna são mostrados os valores de x, na segunda coluna são mostrados os valores de y e na terceira coluna são mostrados os valores de $3e^{-x/2} + x - 2$.

A curva exata e os valores aproximados obtidos são mostradas no gráfico[3]:

[3]gerado através do comando **plot2d([x;x]',[y;3*exp(-x/2)+x-2]',[-3,1],"121",'@aproximado@exato')**



4.4 Método de Runge-Kutta-Fehlberg

Este método implementa procedimentos para verificar se o valor do passo de integração h é adequado. Em cada iteração, são obtidas duas aproximações para a solução $y(x)$. Se as duas respostas satisfazem uma precisão pré-estabelecida, a aproximação é aceita. Se não, o tamanho do passo de integração é reduzido e uma nova iteração é realizada. Se as duas respostas possuem mais digitos significativos do que o requerido, o valor do passo de integração é aumentado e uma nova iteração é realizada.

Cada iteração requer o cálculo dos fatores :

$$\begin{aligned}
f_1 &= hf(x_k, y_k) \\
f_2 &= hf(x_k + \frac{1}{4}h, y_k + \frac{1}{4}f_1) \\
f_3 &= hf(x_k + \frac{3}{8}h, y_k + \frac{3}{32}f_1 + \frac{9}{32}f_2) \\
f_4 &= hf(x_k + \frac{12}{13}h, y_k + \frac{1932}{2197}f_1 - \frac{7200}{2197}f_2 + \frac{7296}{2197}f_3) \\
f_5 &= hf(x_k + h, y_k + \frac{439}{216}f_1 - 8f_2 + \frac{3680}{513}f_3 - \frac{845}{4104}f_4) \\
f_6 &= hf(x_k + \frac{1}{2}h, y_k - \frac{8}{27}f_1 + 2f_2 - \frac{3544}{2565}f_3 + \frac{1859}{4104}f_4 - \frac{11}{40}f_5)
\end{aligned}$$

Uma aproximação para a solução $y(x)$ é obtida pela equação :

$$y_{k+1} = y_k + \frac{25}{216}f_1 + \frac{1408}{2565}f_3 + \frac{2197}{4104}f_4 - \frac{1}{5}f_5$$

A outra aproximação, melhor que a anterior, é obtida através da utilização da equação :

$$z_{k+1} = y_k + \frac{16}{135}f_1 + \frac{6656}{12825}f_3 + \frac{28561}{56430}f_4 - \frac{9}{50}f_5 + \frac{2}{55}f_6$$

Observar que o fator f_2 não aparece diretamente nas duas equações apresentadas.

O passo de integração ótimo, qh, é determinado multiplicando-se o valor de h pelo fator :

$$q = \left(\frac{\delta h}{2\left| z_{k+1} - y_{k+1} \right|} \right)^{1/4}$$

onde δ é a precisão desejada. O valor do passo de integração obedece a relação $h_{min} \leq h \leq h_{max}$. Nesta relação, h_{min} e h_{max} são os limites mínimo e máximo permitidos para a variação do passo de integração.

O método de Runge-Kutta-Fehlberg pode ser implementado utilizando o seguinte algoritmo:

Algorithm 4: Método de Runge-Kutta-Fehlberg

Entrada $[a, b]$, h, $y0$ e δ
Fazer $h_{min} = 0.01$
Fazer $h_{max} = h$
Fazer $x0 = a$
while $x_k < b$ **do**
  Calcular f_1, f_2, f_3, f_4, f_5, e f_6
  Calcular y_{k+1}, z_{k+1} e q
  Calcular $erro = \left| z_{k+1} - y_{k+1} \right| / h$
  if $erro < \delta$ **then**
    Fazer $x_{k+1} = x_k + h$
    $k = k + 1$
  end if
  if $q \leq 0.1$ **then**
    $q = 0.1$
  else if $q \geq 4.0$ **then**
    $q = 4.0$
  end if
  $h = qh$
  if $h > h_{max}$ **then**
    $h = h_{max}$
  end if
  if $x_k + h > b$ **then**
    $h = b - x_k$
  else if $h < h_{min}$ **then**
    Erro; Abortar o processo
  end if
end while
Apresentar valores de x_k e y_k

Vamos considerar a equação diferencial

$$\frac{dy}{dx} = 1 + y^2$$

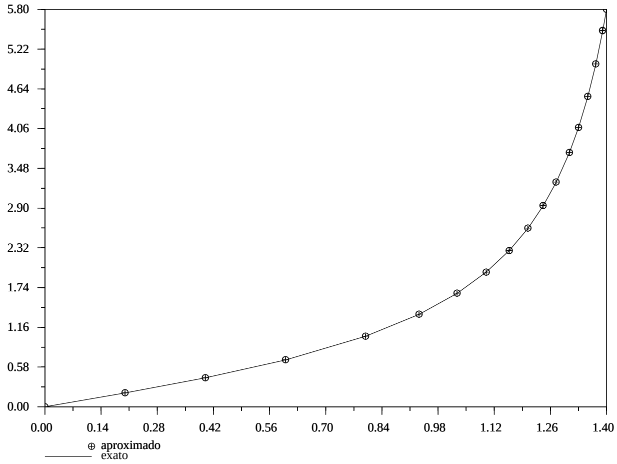
com condição inicial $y(0) = 0$. A solução exata é $y(x) = tan(x)$. Utilizando o método de Runge-Kutta-Fehlberg, obtemos a solução aproximada para $y(x)$ no intervalo $[0, 1.4]$ para um valor inicial de passo de integração, h, igual a 0.2. Nos cálculos, utilizamos $\delta = 2\text{x}10^{-5}$. Usando o Scilab, obtemos os seguintes resultados :

```
-->getf('f3.sci')         // arquivo com a funcao f(x,y) = 1 + y^2

-->getf('rkf.sci')        // metodo de Runge-Kutta-Fehlberg

-->rkf(0, 1.4, 0.2, 0, 0.00002)
 ans  =
!   0.           0.           0.        !
!   0.2          0.2027100    0.2027100 !
!   0.4          0.4227933    0.4227932 !
!   0.6          0.6841376    0.6841368 !
!   0.8          1.0296434    1.0296386 !
!   0.9329059    1.3490448    1.3490363 !
!   1.0274861    1.6558109    1.6557986 !
!   1.1000362    1.9649525    1.9649357 !
!   1.1573791    2.2794841    2.2794621 !
!   1.2038894    2.6021006    2.6020725 !
!   1.2423715    2.9346013    2.9345661 !
!   1.2747137    3.2782032    3.2781598 !
!   1.3072421    3.7060792    3.7060241 !
!   1.3302315    4.0764504    4.0763843 !
!   1.3533276    4.5257236    4.5256426 !
!   1.3733296    4.9982498    4.9981516 !
!   1.3904686    5.4853375    5.4852196 !
!   1.4          5.798015     5.7978837 !

-->
```

Na primeira coluna são mostrados os valores de x, na segunda coluna são mostrados os valores de y e na terceira coluna são mostrados os valores de $tan(x)$.

A curva exata e os valores aproximados obtidos são mostradas no gráfico[4]:

[4]gerado através do comando **plot2d([x;x]',[y;tan(x)]',[-3,1],"121",'@aproximado@exato')**



4.5 Métodos preditores-corretores

Os métodos de Euler, Heun, Runge-Kutta e Runge-Kutta-Fehlberg são chamados de métodos de passo único pois usam apenas informações de um ponto anterior para calcular o próximo.

Os métodos preditores-corretores são métodos de passo múltiplo porque precisam de vários pontos para gerar o próximo. Eles não são métodos auto-inicializáveis, como os anteriores. Precisam de vários pontos iniciais para começarem os cálculos. Estes pontos iniciais são calculados usando os métodos de passo simples.

4.5.1 Método de Adams-Bashforth-Moulton

Consiste em gerar aproximações para a solução de

$$\frac{dy}{dx} = f(x, y)$$

com $y(x_0) = y_0$ no intervalo $[a, b]$ através das equações :

$$p_{k+1} = y_k + \frac{h}{24}[-9f_{k-3} + 37f_{k-2} - 59f_{k-1} + 55f_k]$$

e

$$y_{k+1} = y_k + \frac{h}{24}[f_{k-2} - 5f_{k-1} + 19f_k + 9f_{k+1}]$$

Observar que é necessário o conhecimento prévio dos pontos (x_{k-3}, f_{k-3}), (x_{k-2}, f_{k-2}), $(x_{k-1}, f_{k-1}$ e (x_k, f_k).

Algorithm 5: Método de Adams-Bashforth-Moulton

Entrada $[a, b]$, h e $y0$
Fazer $x0 = a$
Fazer $n = (b - a)/h$
Obter $f_0(x_0, y_0)$, $f_1(x_1, y_1)$, $f_2(x_2, y_2)$ e $f_3(x_3, y_3)$ por RK4.
for k = 3 to n **do**
  Calcular p_{k+1}, x_{k+1}, $f_{k+1}(x_{k+1}, p_{k+1})$
  Calcular y_{k+1}
  Fazer $f_{k-3} = f_{k-2}$, $f_{k-2} = f_{k-1}$ e $f_{k-1} = f_k$
  Calcular $f_k = f(x_{k+1}, y_{k+1})$
end for
Apresentar valores de x_k e y_k

Considerando a equação diferencial,

$$\frac{dy}{dx} = (x - y)/2$$

com condição inicial $y(0) = 1$ e $h = 1/8$, obtemos a solução aproximada para $y(x)$ no intervalo $[0, 3]$. Os resultados apresentados usando o Scilab são :

```
-->getf('f2.sci')        // arquivo com a funcao f(x,y) = (x - y)/2

-->getf('abm.sci')       // Metodo de Adams-Bashforth-Moulton

-->abm(0, 3, 1/8, 1)
 ans  =

!   0.        1.         !
!   0.125     0.9432392 !
!   0.25      0.8974908 !
!   0.375     0.8620874 !
!   0.5       0.8364023 !
!   0.625     0.8198468 !
!   0.75      0.8118677 !
!   0.875     0.8119453 !
!   1.        0.8195917 !
!   1.125     0.8343481 !
!   1.25      0.8557839 !
!   1.375     0.8834943 !
!   1.5       0.9170992 !
!   1.625     0.9562415 !
!   1.75      1.0005856 !
!   1.875     1.0498164 !
!   2.        1.1036378 !
!   2.125     1.1617718 !
```

```
!   2.25     1.2239569 !
!   2.375    1.2899478 !
!   2.5      1.3595139 !
!   2.625    1.4324385 !
!   2.75     1.5085183 !
!   2.875    1.587562  !
!   3.       1.66939   !

-->
```

Na primeira coluna são mostrados os valores de x e na segunda coluna são mostrados os valores de y. Os quatro primeiros pares de valores são aproximações iniciais obtidas através do método de Runge-Kutta de $4^{\underline{a}}$ ordem.

4.5.2 Método de Milne-Simpson

Consiste em gerar aproximações para a solução de

$$\frac{dy}{dx} = f(x,y)$$

com $y(x_0) = y_0$ no intervalo $[a,b]$ através das equações :

$$p_{k+1} = y_{k-3} + \frac{4h}{3}(2f_{k-2} - f_{k-1} + 2f_k)$$

e

$$y_{k+1} = y_{k-1} + \frac{h}{3}(f_{k-1} + 4f_k + f_{k+1})$$

4.5.3 Método de Hamming

Consiste em gerar aproximações para a solução de

$$\frac{dy}{dx} = f(x,y)$$

com $y(x_0) = y_0$ no intervalo $[a,b]$ através das equações :

$$p_{k+1} = y_{k-3} + \frac{44}{3}(2f_{k-2} - f_{k-1} + 2f_k)$$

$$y_{k+1} = \frac{-y_{k-2} + 9y_k}{8} + \frac{3h}{8}(-f_{k-1} + 2f_k + f_{k+1})$$

4.6 Sistemas de Equações Diferenciais

Vamos considerar um sistema de equações diferenciais ordinárias de 1ª ordem,

$$\begin{aligned}
\frac{du}{dx} &= f(x,u,v) \quad \text{com} \quad u(x_0) = u_0 \\
\frac{dv}{dx} &= g(x,u,v) \quad \text{com} \quad v(x_0) = v_0
\end{aligned}$$

Este sistema pode ser resolvido pelos métodos numéricos apresentados anteriormente. Optando pelo método de Runge-Kutta de 4ª ordem, as aproximações para as soluções das equações são geradas através das equações :

$$u_{k+1} = u_k + \frac{h}{6}(f_1 + 2f_2 + 2f_3 + f_4)$$

e

$$v_{k+1} = v_k + \frac{h}{6}(g_1 + 2g_2 + 2g_3 + g_4)$$

com :

$$\begin{aligned}
f_1 &= f(x_k, u_k, v_k) \\
f_2 &= f(x_k + \frac{h}{2}, u_k + \frac{h}{2}f_1, v_k + \frac{h}{2}g_1) \\
f_3 &= f(x_k + \frac{h}{2}, u_k + \frac{h}{2}f_2, v_k + \frac{h}{2}g_2) \\
f_4 &= f(x_k + h, u_k + hf_3, v_k + hg_3)
\end{aligned}$$

e

$$\begin{aligned}
g_1 &= g(x_k, u_k, v_k) \\
g_2 &= g(x_k + \frac{h}{2}, u_k + \frac{h}{2}f_1, v_k + \frac{h}{2}g_1) \\
g_3 &= g(x_k + \frac{h}{2}, u_k + \frac{h}{2}f_2, v_k + \frac{h}{2}g_2) \\
g_4 &= g(x_k + h, u_k + hf_3, v_k + hg_3)
\end{aligned}$$

O método de Runge-Kutta para o sistema acima pode ser implementado através do algoritmo:

Algorithm 6: Método RK4 para sistemas com duas equações diferenciais

Entrada $[a, b]$, h e $u0$, $v0$
Fazer $x0 = a$
Fazer $n = (b - a)/h$
for k = 0 to n **do**
  Calcular f_1, f_2, f_3 e f_4
  Calcular g_1, g_2, g_3 e g_4
  Calcular u_{k+1} e v_{k+1}
  Fazer $x_{k+1} = x_k + h$
end for
Apresentar valores de x_k, u_k e v_k

Vamos considerar o sistema :

$$\begin{aligned} \frac{du}{dx} &= u + 2v \quad \text{com} \quad u(0) = 6 \\ \frac{dv}{dx} &= 3u + 2v \quad \text{com} \quad v(0) = 4 \end{aligned}$$

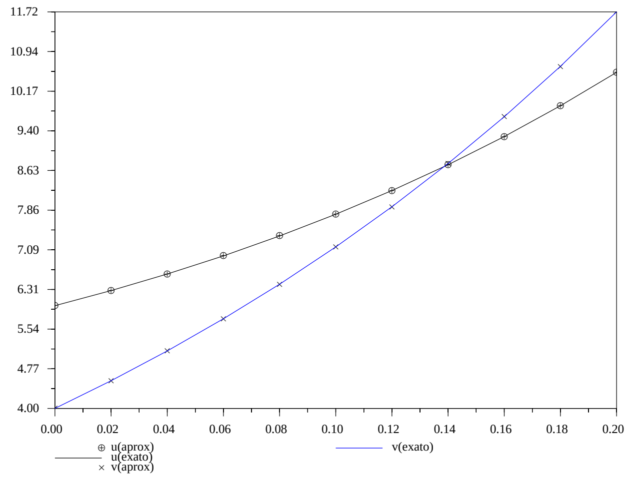
no intervalo $[0, 0.2]$ com passo de integração $h = 0.02$. Utilizando um programa desenvolvido no Scilab, temos :

```
-->getf('f4.sci')       // arquivo com as funcoes f(x,u,v) e g(x,u,v)

-->getf('rk4sis.sci')   //  RK4 para sistemas de 2 equacoes diferenciais

-->rk4sis(0, 0.2, 0.02, 6, 4)
 ans  =

!   0.      6.           4.         !
!   0.02    6.2935455    4.5393249 !
!   0.04    6.6156221    5.119486   !
!   0.06    6.9685253    5.7439653 !
!   0.08    7.3547432    6.416533   !
!   0.1     7.7769729    7.1412722 !
!   0.12    8.2381375    7.9226041 !
!   0.14    8.7414052    8.7653167 !
!   0.16    9.2902095    9.6745954 !
!   0.18    9.8882714    10.656056 !
!   0.2     10.539623    11.715781 !

-->
```

Na primeira coluna são mostrados os valores de x_k, na segunda coluna são mostrados os valores de u_k e na terceira coluna são mostrados os valores de v_k.

A curva exata e os valores aproximados obtidos são mostradas no gráfico[5]:

[5] gerado através do comando **plot2d([x;x;x;x]',[u;4*exp(4*x)+2*exp(-x);v;6*exp(4*x)-2*exp(-x)]',[-3,1,-2,2],"121",'@u(aprox)@u(exato)@v(aprox)@v(exato)')**



Uma equação diferencial ordinária de ordem n pode ser transformada em um sistema de n equações ordinárias de $1^{\underline{a}}$ ordem. Esta transformação é feita através da definição de novas variáveis.

Vamos considerar a equação diferencial de $2^{\underline{a}}$ ordem,

$$\frac{d^2y}{dx^2} - 2\frac{dy}{dx} + 2y = e^{2x} \sin x$$

Definindo duas novas variáveis, $u(x)$ e $v(x)$, como :

$$\begin{aligned} u(x) &= y(x) \\ v(x) &= \frac{dy}{dx} \end{aligned}$$

temos o sistema de equações de $1^{\underline{a}}$ ordem :

$$\begin{cases} \dfrac{du}{dx} = v = f(x,u,v) \\ \dfrac{du_2}{dx} = e^{2x} \sin x - 2u_1(x) + 2u_2(x) = g(x,u,v) \end{cases}$$

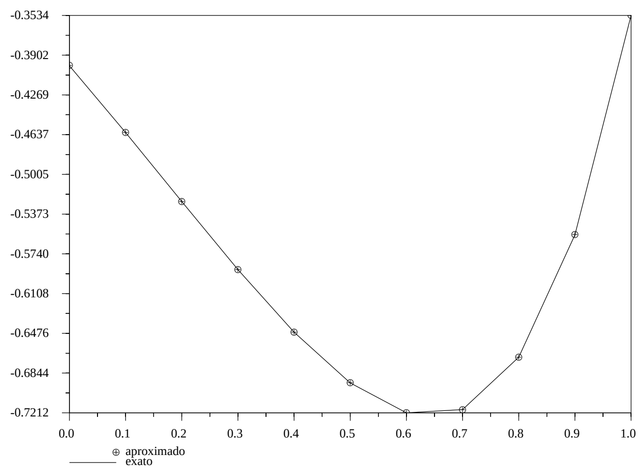
com condições iniciais dadas por $u(0) = -0.4$ e $v(0) = -0.6$. O sistema de equações diferenciais de $1^{\underline{a}}$ ordem pode, então, ser resolvido pelo método de Runge-Kutta. Considerando o intervalo $[0,1]$ e $h = 0.1$, o sistema de equações apresentado é resolvido utilizando o Scilab. Os resultados são :

```
-->getf('f5.sci')        // arquivo com as funcoes f(x,u,v) e g(x,u,v)

-->getf('rk4sis.sci')    // RK4 para sistemas de 2 equacoes diferenciais
```

```
-->rk4sis(0,1, 0.1, -0.4, -0.6)
 ans  =

!   0.   - 0.4         - 0.6       !
!   0.1  - 0.4617333  - 0.6316312 !
!   0.2  - 0.5255599  - 0.6401489 !
!   0.3  - 0.5886014  - 0.6136638 !
!   0.4  - 0.6466123  - 0.5365820 !
!   0.5  - 0.6935667  - 0.3887381 !
!   0.6  - 0.7211519  - 0.1443809 !
!   0.7  - 0.7181530    0.2289970 !
!   0.8  - 0.6697113    0.7719918 !
!   0.9  - 0.5564429    1.5347815 !
!   1.   - 0.3533989    2.5787663 !
```

Na primeira coluna são mostrados os valores de x_k, na segunda coluna são mostrados os valores de $u_k = y_k$ e na terceira coluna são mostrados os valores de $v_k = y'_k$.

A curva exata e os valores aproximados obtidos são mostradas no gráfico[6]:



[6]gerado através do comando **plot2d([x;x]',[y;-exp(2*x).*(2*cos(x) - sin(x))/5]',[-3,1],"121", '@aproximado@exato')**

Referências Bibliográficas

[1] John H. Mathews, Numerical Methods for Mathematics, Science, and Engineering, 2^{nd} Edition, Prentice-Hall, 1992, Capítulo 1

[2] Leônidas C. Barroso e outros, Cálculo Numérico (com aplicações), 2ª Edição, Editora Harbra, 1987, Capítulo 2.

[3] Cristina Cunha, Métodos Numéricos para as Engenharias e Ciências Aplicadas, Editora da Unicamp, 1993, Capítulo 2.

[4] S.D. Conte, Elementos de Análise Numérica, 2ª Edição, Editora Globo, 1975, Capítulo 5.

[5] Runge-Kutta-Fehlberg method, Mathematics 373, Summer 1997, `http://math.rutgers.edu/~ojanen/numa/prog/rkfmain.html`

[6] Richard L. Burden, J. Douglas Faires, Numerical Analysis, 6th Edition, Brooks/Cole Publishing Co., 1997, Capítulo 5.