3 Equações diferenciais

3.1 Forma geral das equações diferenciais

Uma **equação diferencial ordinária** —ou de forma abreviada, EDO— de ordem n é uma relação entre uma função $y(x)$ e as suas derivadas y', y'', ..., $y^{(n)}$. Por exemplo, uma possível equação diferencial ordinária de terceira ordem é:

$$3x^2 y''' + \frac{y}{2y'} = xy'' \tag{3.1}$$

Neste caso, é possível também escrever a equação mostrando explicitamente a expressão que define a derivada de terceira ordem:

$$y''' = \frac{y''}{3x} - \frac{y}{6x^2 y'} \tag{3.2}$$

É conveniente usar uma notação que permite ver a equação duma forma mais geral: a função y será denotada por y_1, y_2 será a primeira derivada y' e y_3 a segunda derivada y''. A equação diferencial é então:

$$y_3' = \frac{y_3}{3x} - \frac{y_1}{6x^2 y_2} \tag{3.3}$$

que define a derivada de y_2 em relação a (x, y_1, y_2, y_3). Para resolver este problema é necessário resolver simultaneamente as equações que definem as derivadas de y_1 e y_2. Ou seja, a equação original (3.1), de terceira ordem, foi transformada no sistema de 3 equações:

$$y_1' = y_2 \tag{3.4}$$
$$y_2' = y_3 \tag{3.5}$$
$$y_3' = \frac{y_3}{3x} - \frac{y_1}{6x^2 y_2} \tag{3.6}$$

Esta forma de escrever a equação diferencial tem também a vantagem de poder ser escrita de forma mais compacta; define-se o vetor,

$$\vec{y} = (y_1, y_2, y_3) \tag{3.7}$$

e o sistema de 3 equações diferenciais é equivalente a uma única equação diferencial vetorial:

$$\frac{\mathrm{d}\vec{y}}{\mathrm{d}x} = \vec{f}(x, \vec{y}) \tag{3.8}$$

onde a derivada do vetor $\vec{y}$ é outro vetor obtido derivando cada uma das suas componentes,

$$\frac{\mathrm{d}\vec{y}}{\mathrm{d}x} = (y_1', y_2', y_3') \tag{3.9}$$

e as três componentes da função vetorial $\vec{f}(x,\vec{y})$ são, neste caso,

$$\vec{f}(x,\vec{y}) = \left(y_2, y_3, \frac{y_3}{3x} - \frac{y_1}{6x^2 y_2} \right) \tag{3.10}$$

Assim sendo, para resolver equações diferencias ordinárias como, por exemplo, a equação (3.1), basta saber resolver equações de primeira ordem com a forma geral

$$\frac{\mathrm{d}y}{\mathrm{d}x} = f(x,y) \tag{3.11}$$

e generalizar o método ao caso em que y e f são vetores com várias componentes.

3.2 Equações diferenciais de primeira ordem

A forma geral das EDO de primeira ordem é

$$y' = f(x,y) \tag{3.12}$$

A função $y(x)$ é chamada **variável dependente** e x é a **variável independente**. Uma solução da EDO, num intervalo $[x_0, x_n]$, é qualquer função y de x que verifica a equação.

Existem em geral muitas soluções, por exemplo, a figura 3.1 mostra 7 possíveis soluções da equação $y' = (x-1)(x-3) - y$. Em cada ponto de cada uma das curvas, o declive tem o mesmo valor obtido substituindo as coordenadas do ponto na expressão $(x-1)(x-3) - y$. As diferentes soluções não se cruzam nunca, porque em cada ponto existe apenas um valor $(x-1)(x-3) - y$ e cada ponto (x,y) pertence unicamente a uma das soluções da equação diferencial.

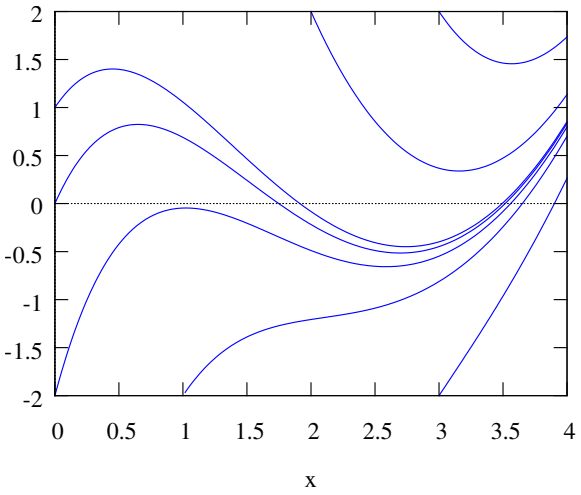

Figura 3.1: Soluções da equação diferencial $y' = (x-1)(x-3) - y$.

Se for dado um valor inicial y_0 para a função y no ponto inicial x_0, existe uma única solução $y(x)$, que é a curva tal que $y(x_0) = y_0$ e com declive igual a $(x-1)(x-3) - y(x)$ em qualquer valor de x.

Os métodos de resolução numérica consistem em calcular o valor da variável dependente y numa sequência discreta de pontos $\{x_0, x_1, x_2, \ldots, x_n\}$, usando alguma aproximação. O resultado obtido aplicando um determinado método será o conjunto de pontos $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$, que aproximam o gráfico da função $y(x)$. Nos métodos que usam incrementos constantes, o intervalo $[x_0, x_n]$ divide-se em n subintervalos de comprimento $h = (x_n - x_0)/n$, de forma que cada valor na sequência $\{x_i\}$ é igual ao valor anterior mais h:

$$\{x_0, x_1 = x_0 + h, x_2 = x_0 + 2h, \ldots, x_n = x_0 + nh\} \tag{3.13}$$

Por exemplo, a figura 3.2 mostra a solução da equação diferencial $y' = f(x,y)$ com $f = (x-1)(x-3) - y$ e condição inicial $y(0) = 1$, no intervalo $0 \leq x \leq 4$. Os cinco pontos sobre a curva são a aproximação da função usando apenas 5 pontos com abcissas 0, 1, 2, 3 e 4.

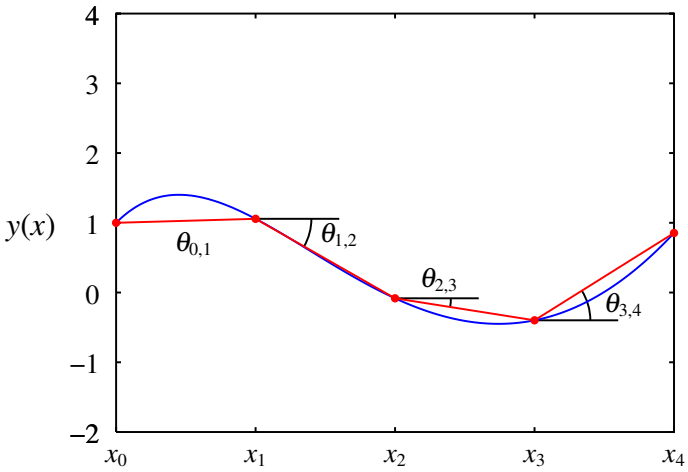

Figura 3.2: Uma solução da equação $y' = (x-1)(x-3) - y$, e aproximação com 5 pontos.

Cada ângulo $\theta_{i,j}$ na figura é a inclinação segmento entre os pontos (x_i, y_i) e $(x_{i,j}, y_{i,j})$. A tangente do ângulo $\theta_{i,j}$ (declive médio) é igual ao valor médio de $f(x,y)$ no intervalo $[x_i, x_j]$. A partir das coordenadas (x_i, y_i) de um dos pontos, o declive médio $\bar{f}_{i,i+1}$ permite calcular as coordenadas do ponto seguinte:

$$\boxed{x_{i+1} = x_i + h \qquad y_{i+1} = y_i + h\,\bar{f}_{i,i+1}} \tag{3.14}$$

Usando estas **relações de recorrência** de forma iterativa, consegue-se obter as coordenadas de todos os pontos (x_i, y_i), começando com os valores iniciais dados (x_0, y_0).

Se os declives médios $\bar{f}_{i,j}$ pudessem ser calculados de forma exata, os resultados obtidos seriam exatos. No entanto, para calcular o valor médio de $f(x,y)$ num intervalo seria

necessário conhecer a função $y(x)$, mas quando se usam métodos numéricos é porque não existem métodos para encontrar a expressão analítica dessa função. Os diferentes métodos numéricos que existem para resolver equações diferenciais correspondem a diferentes esquemas para estimar o valor médio aproximado da função $f(x,y)$ em cada intervalo.

3.2.1 Método de Euler

No método de Euler admite-se que $\bar{f}_{i,i+1} \approx f(x_i, y_i)$. Ou seja, o valor médio de f em cada intervalo é aproximado pelo valor de $f(x_i, y_i)$ no ponto inicial do intervalo. Realizando os cálculos no Maxima para o caso considerado acima, em que $f(x,y) = (x-1)(x-3) - y$ e com condição inicial $y(0) = 1$, pode usar-se uma lista p para armazenar as coordenadas dos pontos; inicia-se a lista com o ponto inicial e define-se a função $f(x,y)$ dada:

```
(%i1) p: [[0,1]]$
(%i2) f(x,y) := (x-1)*(x-3)-y$
```

Os quatro pontos seguintes na sequência são acrescentados usando um ciclo e usando as relações de recorrência (3.14) com $h = 1$:

```
(%i3) for i thru 4 do
        ([xi,yi]: last(p),
         p: endcons ([xi + 1, yi + f(xi, yi)], p))$
(%i4) p;
(%o4)        [[0, 1], [1, 3], [2, 0], [3, - 1], [4, 0]]
```

O gráfico na figura 3.3 mostra o resultado obtido (segmentos de reta e pontos), comparando-o com a solução exata (curva contínua). Observa-se que a solução obtida com o método de Euler não é uma aproximação muito boa. No entanto, reduzindo o valor dos incrementos em x de $h = 1$ para um valor menor, deverá ser possível obter uma aproximação melhor. É conveniente reduzir gradualmente o valor de h e comparar as soluções com as obtidas com valores maiores de h; se a solução obtida não variar significativamente quando h é reduzido, essa solução deverá ser uma boa aproximação da solução verdadeira.

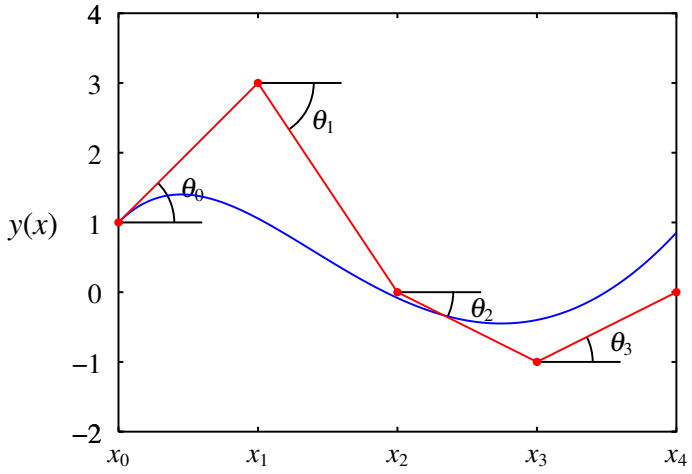

Figura 3.3: Método de Euler.

Exemplo 3.1
A carga armazenada num condensador ligado a uma pilha e a uma resistência é uma função Q que depende do tempo t e verifica a seguinte equação diferencial

$$\frac{dQ}{dt} = 1.25 - Q$$

No instante inicial, $t = 0$, a carga no condensador é nula. Usando o método de Euler, encontre uma sequência que aproxime $Q(t)$ no intervalo $0 \leq t \leq 6$.

Resolução. Neste caso, a função f que define a derivada não depende da variável independente t. Os valores iniciais e a função que define a derivada são:

```
(%i5) p: [[0,0]]$
(%i6) f(Q) := 1.25 - Q$
```

Começando com incrementos de tempo $h = 0.1$, são necessárias 60 iterações para terminar em $t = 6$.

```
(%i7) for i thru 60 do
        ([t0,Q0]: last(p),
         p: endcons ([t0 + 0.1, Q0 + 0.1*f(Q0)], p))$
```

O último ponto na lista p é:

```
(%i8) last(p);
(%o8)          [5.999999999999995, 1.247753737125107]
```

Convém repetir o processo, com um valor menor de h, por exemplo 0.01 e comparar o novo resultado com o resultado anterior:

```
(%i9) q: [[0,0]]$
(%i10) for i thru 600 do
         ([t0,Q0]: last(q),
          q: endcons ([t0 + 0.01, Q0 + 0.01*f(Q0)], q))$
(%i11) last(q);
(%o11)         [5.999999999999917, 1.246993738385861]
```

A pesar de que o resultado final coincide em 4 algarismos significativos nos dois casos, convém comparar as duas sequências completas. As duas soluções obtidas com $h = 0.1$ e $h = 0.01$ estão armazenadas nas listas p e q e podem ser representadas num gráfico. A figura 3.4 mostra que a discrepância entre as duas soluções obtidas não é muito grande, concluindo-se que a solução com $h = 0.01$ deverá estar já bastante próxima da solução verdadeira. O gráfico foi obtido usando o seguinte comando

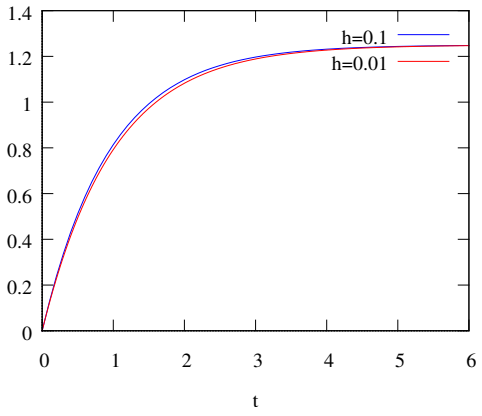

Figura 3.4: Soluções pelo método de Euler.

```
(%i12) plot2d([[discrete, p],[discrete, q]],
               [xlabel, "t"],[legend, "h=0.1", "h=0.01"])$
```

3.2.2 Método de Euler melhorado

A maior discrepância entre a sequência de pontos que aproximam a solução exata $y(x)$ na figura 3.2 e a sequência obtida usando o método de Euler, apresentada na figura 3.3, é no segundo ponto, (x_1, y_1). O valor médio do declive no primeiro intervalo, que era aproximadamente igual a zero, no método de Euler foi substituído por 2, que é o declive no ponto inicial (x_0, y_0). A figura 3.2 mostra que mantendo um valor fixo $y = y_0 = 1$, o declive $f(x,y)$ muda de um valor positivo em $x_0 = 0$ para um valor negativo em $x_1 = 1$. Se o valor do declive médio fosse aproximado pela média entre esses dois declives, o resultado seria muito melhor.

O método de Euler melhorado consiste em admitir que o declive médio $\bar{f}_{i,i+1}$ é igual à média entre os declives no ponto inicial (x_i, y_i) e no ponto final (x_{i+1}, y_{i+1}):

$$\boxed{\bar{f}_{i,i+1} \approx \frac{f(x_i,y_i)+f(x_{i+1},y_{i+1})}{2}} \tag{3.15}$$

O problema com esta equação é que para poder calcular $f(x_{i+1},y_{i+1})$ seria necessário saber o valor de y_{i+1}, mas esse valor é desconhecido enquanto a solução da equação não for conhecida. É possível fazer uma estimativa inicial do valor que y_{i+1} poderá ter e a seguir melhorar essa estimativa. Uma primeira estimativa muito rudimentar consiste em dizer que y_{i+1} é igual a y_i. Com essa abordagem, a resolução da mesma equação $y' = (x-1)(x-3) - y$ considerada na secção anterior, com valor inicial $y(0) = 1$ e com 5 pontos no intervalo $0 \le x \le 4$, pode ser feita assim: começa-se por iniciar a lista de pontos e definir a função para o declive

```
(%i13) p: [[0,1]]$
(%i14) f(x,y) := (x-1)*(x-3)-y$
```

Foi necessário definir novamente a função $f(x,y)$, pois a definição que foi inserida na secção anterior foi logo substituída pela função $f(Q)$ do exemplo 3.1. Basta repetir as 4 iterações mas usando agora a expressão (3.15) nas relações de recorrência (3.14)

```
(%i15) for i thru 4 do
         ([xi,yi]: last(p),
          p: endcons ([xi+1, yi + (f(xi,yi)+f(xi+1,yi))/2], p))$
(%i16) p;
                        3          1          1        3
(%o16)     [[0, 1], [1, -], [2, - -], [3, - -], [4, -]]
                        2          2          2        2
```

A figura 3.5 mostra a sequência de pontos obtida, juntamente com a solução exata (curva contínua na figura). O resultado é muito melhor que o resultado obtido com o método de Euler. Este resultado pode ser melhorado se a estimativa inicial $y_{i+1} = y_i$ for melhorada com o resultado obtido a partir dessa estimativa rudimentar, como se explica a seguir.

A partir do valor inicial y_0, começa-se por fazer a primeira estimativa do valor de y_1, usando $y_{1,0} = y_0$, como nos cálculos acima, e calcula-se $y_{1,1} = y_0 + h f(x_0, y_{1,0})$. Esse

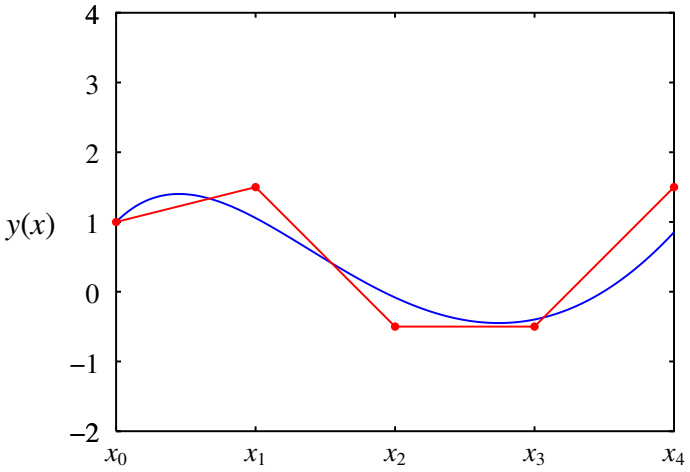

Figura 3.5: Solução exata e solução obtida com o método usado em (%i15).

valor estará mais próximo do valor real de y_1 do que a estimativa inicial $y_{1,0}$; assim sendo, espera-se que a sequência $y_{1,n} = y_0 + h\,f(x_0, y_{1,n-1})$ converja para um valor mais próximo do valor real y_1. Calculam-se alguns termos dessa sequência, até $|y_{1,n} - y_{1,n-1}|$ ser menor que uma precisão desejada. Nesse momento usa-se $y_1 = y_{1,n}$ como valor inicial para o seguinte intervalo e o processo repete-se iterativamente em todo o intervalo de x. No caso do problema resolvido acima, com precisão de 0.0001, o método pode ser implementado assim:

```
(%i17) f(x,y) := float ((x-1)*(x-3)-y)$
(%i18) p: [[0,1]]$
(%i19) for i thru 4 do
         ([xi,yi]: last(p),
          y1: yi,
          y2: yi + (f(xi,yi)+f(xi+1,y1))/2,
          while abs(y1-y2) > 0.0001 do
                (y1: y2, y2: yi + (f(xi,yi)+f(xi+1,y1))/2),
          p: endcons ([xi+1, y2], p))$
(%i20) p;
(%o20) [[0, 1], [1, 1.33331298828125], [2, .1111229788511992],
                  [3, - .2962674737252655], [4, .9012259028472571]]
```

A função f foi definida novamente, usando-se o comando `float`, para evitar obterr fracções com numeradores e denominadores muito grandes; nos exemplos anteriores foram feitos menos cálculos e, por isso, não produziram frações com números grandes. O resultado (figura 3.6) mostra que a solução obtida é muito melhor do que nas figuras 3.3 e 3.5, em relação à solução real.

A pesar de se ter usado uma precisão de 0.0001, não se pode garantir que a solução obtida esteja dentro dessa precisão em relação á solução real, já que a equação 3.15 é apenas uma aproximação e não o valor médio real da derivada. A aproximação pode ser melhorada se em vez de se fazer a média do declive em dois pontos, o declive fosse calculado em

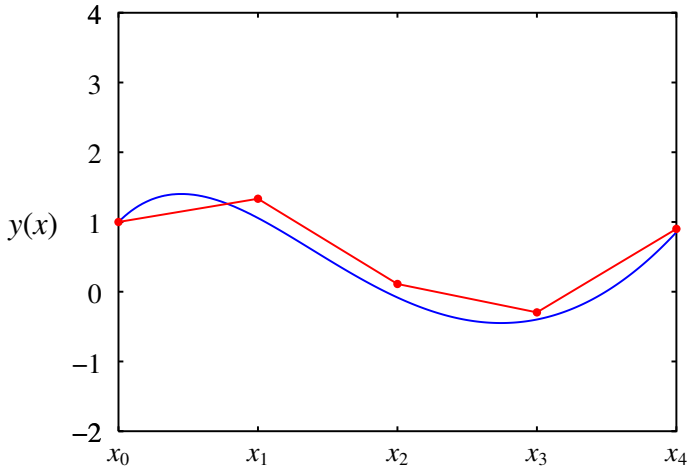

Figura 3.6: Solução exata e solução obtida com o método de Euler melhorado.

outros pontos nessa vizinhança e fosse feita uma média com diferentes pesos que podem ser definidos de forma a minimizar o erro. Existem muitos métodos diferentes que usam esse esquema; o mais popular é o que será descrito na seguinte secção.

3.2.3 Método de Runge-Kutta de quarta ordem

Neste método o valor médio do declive $\bar{f}_{i,i+1}$ obtém-se a partir da média dos declives em 4 pontos diferentes, com pesos diferentes (figura 3.7).

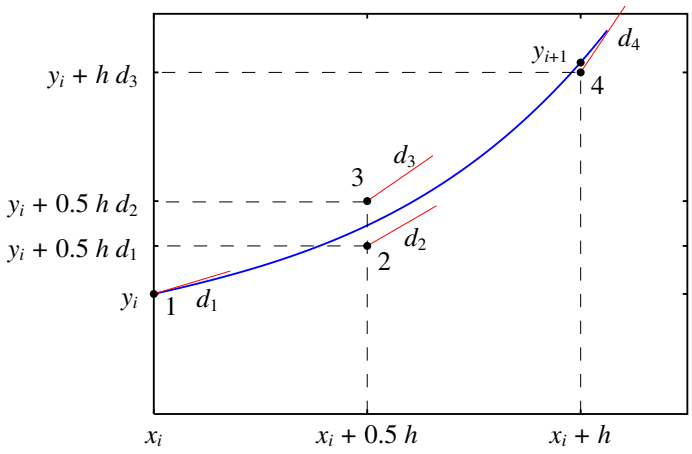

Figura 3.7: Os declives usados no método de Runge-Kutta.

Começa-se por calcular o declive no ponto inicial do intervalo, tal como no método de Euler:

$$d_1 = f(x_i, y_i) \tag{3.16}$$

a seguir, realiza-se um deslocamento na direção desse declive, avançando-se uma distância

$h/2$ no eixo das abcissas, até um ponto 2 (ver figura 3.7). Nesse ponto 2, calcula-se um segundo valor do declive

$$d_2 = f(x_i + h/2, y_i + (h/2)d_1) \tag{3.17}$$

Esse novo valor do declive é usado novamente, para realizar outro deslocamento a partir do ponto inicial, avançando $h/2$ na direção do eixo das abcissas, até um outro ponto 3, onde é calculado um terceiro valor do declive

$$d_3 = f(x_i + h/2, y_i + (h/2)d_2) \tag{3.18}$$

seguindo a direção desse declive d_3, realiza-se um terceiro deslocamento, a partir do ponto inicial, desta vez avançando uma distância h no sentido do eixo das abcissas, para chegar até um ponto 4, onde se calcula um quarto valor do declive

$$d_4 = f(x_i + h, y_i + h\,d_3) \tag{3.19}$$

Pode mostrar-se que para minimizar o erro cometido, o valor médio do declive deve ser aproximado pela seguinte combinação linear dos quatro declives calculados:

$$\bar{f}_{i,i+1} \approx \frac{1}{6}(d_1 + 2d_2 + 2d_3 + d_4) \tag{3.20}$$

no exemplo da figura 3.7, esse valor médio da derivada desloca o ponto inicial até o ponto 4, que está bastante próximo da solução exata da equação.

Para aplicar este método à mesma equação $y' = (x-1)(x-3) - y$ considerada nas secções anteriores, com valor inicial $y(0) = 1$ e com 5 pontos no intervalo $0 \le x \le 4$, inicia-se a lista de pontos e define-se a função para o declive:

```
(%i21) p: [[0,1]]$
(%i22) h: 1$
```

Neste caso foi dado explicitamente o valor do comprimento dos intervalos, para que as expressões seguintes sejam mais claras.

A seguir realizam-se as 4 iterações necessárias para chegar até o ponto $x = 4$. Em cada iteração calculam-se os quatro declives d_1, d_2, d_3 e d_4 e a média com os pesos usados neste método.

```
(%i23) for i thru 4 do
         ([xi,yi]: last(p),
          d1: f(xi, yi),
          d2: f(xi+h/2, yi+(h/2)*d1),
          d3: f(xi+h/2, yi+(h/2)*d2),
          d4: f(xi+h, yi+h*d3),
          p: endcons ([xi+h, yi + h*(d1+2*d2+2*d3+d4)/6], p))$
(%i24) p;
(%o24) [[0, 1], [1, 1.020833333333333], [2, -.09635416666666652],
                  [3, -.3902994791666666], [4, .8744710286458335]]
```

A figura 3.8 mostra a sequência de pontos obtida, juntamente com a solução exata (curva contínua na figura). O resultado é bastante bom, a pesar do valor tão elevado que foi usado para os incrementos da variável x. No caso da equação diferencial resolvida neste exemplo, existem métodos analíticos que permitem encontrar a expressão para a curva contínua que foi apresentada na figura; nos casos em que não é possível encontrar a solução analítica, o método de Runge-Kutta de quarta ordem é uma boa opção para encontrar uma boa aproximação numérica.

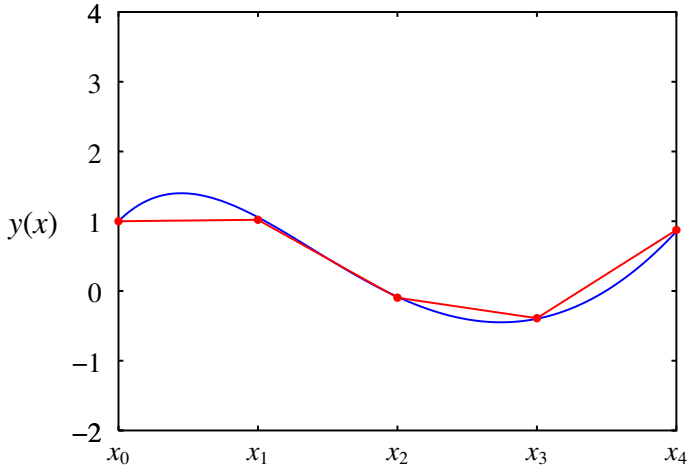

Figura 3.8: Solução exata e solução obtida com o método de Runge-Kutta.

Já existe uma função `rk` predefinida no Maxima, que executa o método de Runge-Kutta de quarta ordem. No exemplo acima, bastava indicar a expressão de $f(x,y)$, o nome que identifica a variável independente, o seu valor inicial e o intervalo em que se quer obter a solução, incluindo o valor dos incrementos h. Nomeadamente, a mesma lista obtida com os comandos (%i17) até (%i21) podia ser obtida com um único comando:

```
(%i25) rk ((x-1)*(x-3)-y, y, 1, [x,0,4,1]);
(%o25) [[0.0, 1.0], [1.0, 1.020833333333333],
[2.0, - .09635416666666652], [3.0, - .3902994791666666],
[4.0, .8744710286458335]]
```

3.3 Sistemas de equações diferenciais

Os métodos descritos nas secções anteriores podem ser generalizados facilmente a um sistema de várias equações, quando sejam conhecidas condições iniciais para todas as variáveis. Tal como se explica no inicio do capítulo, basta admitir que y é um vetor e que f é uma função vetorial.

Exemplo 3.2
Usando o método de Runge-Kutta de quarta ordem, encontre a solução do seguinte sistema de equações diferenciais:

$$\begin{cases} \dfrac{dx}{dt} = 4 - x^2 - 4y^2 \\ \dfrac{dy}{dt} = y^2 - x^2 + 1 \end{cases}$$

no intervalo, $0 \leq t \leq 4$, com condições iniciais $x_0 = -1.25$e $y_0 = 0.75$, em $t_0 = 0$.

Resolução. A variável independente é t e a variável dependente é o vetor $\vec{r} = (x, y)$, que no Maxima é representado como uma lista com duas partes, [x, y]. A expressão $\vec{f}$ que define as derivadas das variáveis dependentes também será uma lista com duas partes, $[4 - x^2 - 4y^2, y^2 - x^2 + 1]$, mas é mais conveniente (para poder generalizar ao caso de n variáveis) representá-la em função das componentes da lista r:

```
(%i26) f(t,r) := [4-r[1]^2-4*r[2]^2, r[2]^2-r[1]^2+1]$
```

foi dada também a variável independente t como argumento da função, a pesar de que neste caso $\vec{f}$ não depende de t, para que os comandos usados a seguir possam ser reutilizados em casos mais gerais.

Com as condições iniciais $t_0 = 0$ e $\vec{r}_0 = (-1.25, 0.75)$ cria-se a lista onde serão inseridos os resultados das iterações

```
(%i27) p: [[0,-1.25,0.75]]$
```

Observe-se que a lista inicial [0,-1.25,0.75] e todas as outras listas que serão acrescentadas, incluem a variável independente t e o vetor dependente $\vec{r}$. Para extrair apenas a variável independente t usa-se o comando `first`, que extrai o primeiro elemento de uma lista e para extrair o vector $\vec{r}$ usa-se o comando `rest`, que produz uma nova lista excluindo o primeiro elemento da lista original; isto é, rest([0, -1.25, 0.75]) produz [-1.25, 0.75].

Usando incrementos $h = 0.02$ para a variável independente, será necessário realizar 200 iterações para terminar em $t = 4$.

```
(%i28) h: 0.02$
(%i29) for i thru 200 do
         (ti: first (last(p)),
         ri: rest (last(p)),
         d1: f (ti, ri),
         d2: f (ti+h/2, ri+(h/2)*d1),
         d3: f (ti+h/2, ri+(h/2)*d2),
         d4: f (ti+h, ri+h*d3),
         p: endcons (cons (ti+h, ri+h*(d1+2*d2+2*d3+d4)/6), p))$
```

O comando `cons` é semelhante a `endcons`, mas insere um elemento no início de uma lista, em vez de no fim. Neste caso `cons` foi usado para inserir o valor de t na lista [x, y],

produzindo a lista [t, x, y], que foi logo inserida no fima da lista p dos resultados.

O último ponto na solução obtida é:

```
(%i30) last(p);
(%o30) [4.000000000000003, 1.232365393486131, - .7493152366008236]
```

O comando `rk` também pode ser usado para obter o mesmo resultado:

```
(%i31) p:rk([4-x^2-4*y^2,y^2-x^2+1],[x,y],[-1.25,0.75],[t,0,4,h])$
```

Para imprimir os resultados é conveniente usar o comando `printf`. Esse comando aceita um formato que diz como devem ser imprimidos os dados. Neste exemplo, usando o formato `"~{~{~f~^, ~}~%~}"` conseguem-se imprimir os 3 elementos t, x e y de cada iteração numa linha separada, separados por uma vírgula e um espaço. Os dois símbolos `~{` e `~}` usam-se para delimitar um formato que deve ser aplicado recursivamente a todos os elementos de uma lista, neste caso a lista p. Dentro desses símbolos, o formato `"~{~f~^, ~}~%"` determina que o formato `"~f~^, "` seja aplicado novamente a cada elemento das sublistas de p, neste caso a t_i, x_i e y_i, e a seguir se insira um fim de linha (símbolo `~%`). O símbolo `~f` é usado para escrever números de vírgula flutuante e o símbolo `~^` indica que o que vem a seguir só será escrito se não se tiver chegado ao último elemento; ou seja, só será escrita uma vírgula após t_i e x_i, mas não após y_i.

Para imprimir unicamente os 9 resultados para $t = 0, 0.5, \ldots, 4$, extraem-se unicamente esses elementos da lista p, usando o comando `makelist`:

```
(%i32) printf(true,"~{~{~f~^, ~}~%~}", makelist(p[25*i+1],i,0,8))$
0.0, -1.25, 0.75
0.5, -1.1773384444565893, 0.8294212227394446
1.0, -1.5811924578377141, 0.818421880110717
1.5, -1.5173593339612323, 0.037780513984502606
2.0, -0.0005587219405601279, 0.06781851774672519
2.5, 1.4685416405394016, 0.14800909710023502
3.0, 1.6981292199163442, -0.7368848754678627
3.5, 1.188041502209304, -0.8582807863663763
4.0, 1.232365393486131, -0.7493152366008236
```

Finalmente, pode ser conveniente gravar esses resultados num ficheiro. Por exemplo, os programas de folha de cálculo conseguem ler dados como estão apresentados acima, dentro de ficheiros do tipo CSV (*Comma Separated Values*). Para gravar o que é apresentado no ecrã num ficheiro, usa-se o comando `with_stdout`

```
(%i33) with_stdout ("resultados.csv",
       printf(true,"~{~{~f~^, ~}~%~}", makelist(p[25*i+1],i,0,8)))$
```

Os resultados ficam gravados no ficheiro resultados.csv.