1 Raízes de equações

1.1 Equações transcendentes

Algumas equações que dependem de uma variável x podem ser resolvidas para obter um ou mais valores de x que verificam a equação. Por exemplo, a equação $x^2+4x+3=0$ tem duas soluções, $x=1$ e $x=3$, e a equação $\mathrm{e}^x=3$ tem uma única solução $x=\ln(3)=1.0986\ldots$ Existem equações, chamadas **transcendentes**, em que não é possível escrever uma expressão matemática para a solução x e até pode ser difícil determinar se existem soluções e quantas. Por exemplo, a equação $x+1=\tan(x)$ é uma equação transcendente; as suas soluções só podem ser calculadas de forma aproximada, usando métodos numéricos.

Para facilitar a procura das soluções de uma equação transcendente, é conveniente rescrevê-la na forma $f(x)=0$; por exemplo, a equação $x+1=\tan(x)$ pode ser escrita como $x+1-\tan(x)=0$. O problema de encontrar as soluções consiste então em encontrar as **raízes** da função $f(x)$, ou seja, os pontos onde o gráfico de $f(x)$ corta o eixo das abcissas. No exemplo anterior, $f(x)=x+1-\tan(x)$, o gráfico da função no intervalo $0\leq x\leq 2\pi$ obtém-se no Maxima com o seguinte comando:

```
(%i1) plot2d(x+1-tan(x),[x,0,2*%pi],[y,-5,5],[ylabel,"f(x)"]);
```

e o resultado (ver figura 1.1) mostra que nesse intervalo existem duas soluções, próximas dos valores $x=1.1$ e $x=4.5$.

No caso das funções contínuas, se em dois pontos diferentes a e b (admitindo $a<b$), os sinais de $f(a)$ e $f(b)$ são diferentes, deve existir pelo menos uma raiz de $f(x)$ no intervalo $a<x<b$. No caso da figura 1.1, vê-se que $f(0)$ é positiva mas $f(1.3)$ é negativa; assim sendo, existe pelo menos uma raiz entre 0 e 1.3. Em $x=3$, $f(3)$ é positiva, mas comparando com o valor negativo de $f(1.3)$ não se pode concluir que existam raízes no intervalo $1.3<x<3$, já que nesse intervalo a função não é contínua.

1.2 Método de bissecções

Dados dois valores diferentes x_1 e x_2, onde $x_1<x_2$ e $f(x_1)\,f(x_2)<0$ (ou seja, os sinais de $f(x_1)$ e $f(x_2)$ são diferentes), calculam-se o ponto meio $x_m=(x_1+x_2)/2$ e o valor da função nesse ponto, $f(x_m)$. Se $f(x_m)$ for nula, x_m é a raiz procurada; caso contrário, se o sinal de $f(x_m)$ for o mesmo de $f(x_1)$ substitui-se x_1 por x_m, se o sinal de $f(x_m)$ for o mesmo de $f(x_2)$ substitui-se x_2 por x_m e o processo repete-se indefinidamente até se obter

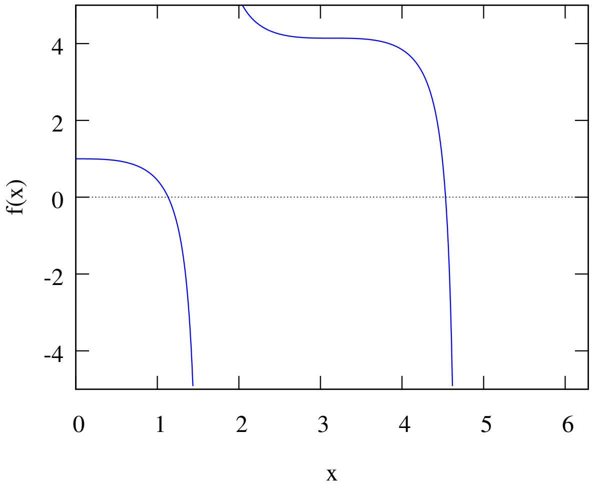

Figura 1.1: Gráfico da função $x+1-\tan(x)$.

um intervalo de comprimento muito reduzido:

$$|x_2 - x_1| < \varepsilon \tag{1.1}$$

onde $\varepsilon/2$ é a precisão com que se quer calcular a raiz. O valor final de x_m será o que melhor aproxima o valor da raiz, com a precisão desejada.

> **Exemplo 1.1**
> Calcule a raiz de $f(x) = x+1-\tan(x)$ no intervalo $0 < x < 1.3$ com 2 casas decimais, usando o método de bisseções.

Resolução. Definem-se primeiro a função e os pontos iniciais:

```
(%i2) f(x) := float (x+1-tan(x))$
(%i3) [x1, x2]: [1, 1.2]$
```

É importante usar `float`, para garantir que a função dê sempre um número e não uma expressão.

Os valores da função nos pontos iniciais são

```
(%i4) [f(x1), f(x2)];
(%o4)         [.4425922753450977, - .3721516221263186]
```

e como os sinais são diferentes, é possível usar o método das bissecções. O ponto meio e o valor da função nesse ponto são:

```
(%i5) [xm: (x1+x2)/2, f(xm)];
(%o5)                  [1.1, .1352403427513478]
```

Como o sinal da função nesse ponto é positivo, substitui-se x_1 pelo ponto meio e repete-se o passo anterior:

```
(%i6) x1: xm$
(%i7) [xm: (x1+x2)/2, f(xm)];
(%o7)                [1.15, - .08449694875532554]
```

Como agora a função é negativa, substitui-se x_2 por x_m e repete-se o mesmo procedimento até se observar que o ponto meio não mude, até à segunda casa decimal:

```
(%i8) x2: xm$
(%i9) [xm: (x1+x2)/2, f(xm)];
(%o9)                 [1.125, .03242872362782112]
(%i10) x1: xm$
(%i11) [xm: (x1+x2)/2, f(xm)];
(%o11)              [1.1375, - .02411658214848611]
(%i12) x2: xm$
(%i13) [xm: (x1+x2)/2, f(xm)];
(%o13)              [1.13125, 0.00461493349083808]
```

O valor da raiz é $x = 1.13$. Há que ter muita atenção a qual dos valores, x_1 ou x_2, deve ser substituído por x_m, para garantir que a raiz esteja sempre dentro do intervalo atual.

As iterações também podiam ter sido feitas mais rapidamente usando um ciclo `while`. Por exemplo, para encontrar a raiz com 3 casas decimais (precisão igual a 0.0005), começando com os mesmos valores iniciais, pode escrever-se:

```
(%i14) [x1, x2]: [1.1, 1.2]$
(%i15) while abs(x2-x1) > 0.001 do
  (xm:(x1+x2)/2, if f(x1)*f(xm) > 0 then x1:xm else x2:xm, print(xm))$
1.15
1.125
1.1375
1.13125
1.134375
1.1328125
1.13203125
```

A raiz, com precisão de 3 casas decimais é 1.132. Observe-se que a função já tinha sido definida previamente e que já estava garantido que os sinais da função são diferentes nos dois pontos iniciais.

1.3 Método de falsa posição

No método das bissecções, se um dos valores $f(x_1)$ ou $f(x_2)$ estiver muito mais próximo de zero, espera-se que a raiz também esteja mas próxima do ponto x_1 ou x_2 em que a função está mais próxima de zero. Como tal, será mais eficiente usar o ponto x_r em que a reta que passa pelos pontos $(x_1, f(x_1))$ e $(x_2, f(x_2))$ intersecta o eixo dos x. Uma relação geométrica de semelhança entre triângulos permite demonstrar que x_r é dado pela expressão

$$x_r = x_2 - f(x_2)\frac{x_2 - x_1}{f(x_2) - f(x_1)} \tag{1.2}$$

O método de falsa posição é semelhante ao método de bissecções, excepto que em vez de se usar o ponto meio de cada intervalo usa-se o ponto x_r definido pela equação anterior.

Exemplo 1.2
Calcule a raiz de $f(x) = x + 1 - \tan(x)$ no intervalo $0 < x < 1.3$ com 3 casas decimais, usando o método de falsa posição.

Resolução. Aproveitando que a função é a mesma que já foi definida no exemplo da secção anterior, basta definir novamente os valores iniciais e modificar o ciclo `while` usado na secção anterior. É conveniente também guardar os valores da função já calculados, para evitar a redundância de serem calculados novamente a cada iteração:

```
(%i16) [x1, x2]: [1.1, 1.2]$
(%i17) [f1, f2]: [f(x1), f(x2)];
(%o17)        [.1352403427513478, - .3721516221263186]
```

Os valores da função nos dois extremos do intervalo são armazenados nas variáveis f_1 e f_2; o resultado (`%o17`) permite também conferir que o sinal da função muda entre x_1 e x_2. Como será claro nos resultados no fim deste exemplo, neste método a distância entre x_1 e x_2 não tem que ser pequena para que o valor x_r esteja muito próximo da raiz. Assim sendo, a condição de paragem $|x_2 - x_1| < \varepsilon$ já não serve e no seu lugar deve comparar-se cada valor x_r com o que tenha sido obtido na iteração anterior, que representaremos por x_0. Inicialmente pode admitir-se que x_0 e x_r são os próprios x_1 e x_2.

```
(%i18) [x0, xr]: [x1, x2]$
```

Para obter a precisão de 3 casas decimais, usa-se o comando:

```
(%i19) while abs(xr-x0) > 0.0005 do
  (x0:xr, xr:x2-f2*(x2-x1)/(f2-f1), fr:f(xr),
  print (x1, x2, xr),
  if f1*fr>0 then (x1:xr,f1:fr) else (x2:xr,f2:fr))$
1.1 1.2 1.126654017428903
1.126654017428903 1.2 1.131297851469829
1.131297851469829 1.2 1.132100363591035
1.132100363591035 1.2 1.132238851322909
```

Note-se que o valor da raiz com três casas decimais, 1.132, foi obtido na quarta iteração, enquanto que no método de bisseções foram precisas 7 iterações. Neste caso foram também apresentados os valores de x_1 e x_2 em cada iteração, para mostrar que o valor de x_2 permanece sempre em 1.2 e, como tal, $|x_2 - x_1|$ não diminui muito.

1.4 Método de aproximações sucessivas

Quando a equação $f(x) = 0$ pode ser escrita na forma,

$$x = g(x) \tag{1.3}$$

o método de aproximações sucessivas consiste em começar com um valor inicial x_0 e calcular a sequência $x_1 = g(x_0)$, $x_2 = g(x_2)$, ... Dependendo da função g e do valor inicial, em alguns casos a sequência aproxima-se de um limite, isto é, dentro de uma tolerância numérica dada, o valor de $g(x_n)$ será igual a x_n a partir de algum inteiro n e, nesse caso x_n será raiz de $f(x)$.

Exemplo 1.3
Encontre a raiz de $f(x) = x + 1 - \tan(x)$ mais próxima de $x = 1.1$, com 3 casas decimais, usando o método de aproximações sucessivas.

Resolução. A equação $x + 1 - \tan(x) = 0$ pode também ser escrita:

$$x = \tan(x) - 1$$

Definindo a função $g(x) = \tan(x) - 1$ e com valor inicial $x_0 = 1.1$, os seguintes 6 valores na sequência x_i são:

```
(%i20) g(x) := float (tan(x)-1)$
(%i21) xi: 1.1$
(%i22) for i:1 thru 6 do (xi: g(xi), print (xi))$
.9647596572486523
.4429269858864537
- .5256388221063686
- 1.580073542507877
106.7878688889125
- 1.026287385725812
```

Esta sequência não parece ser convergente. O problema neste caso é que uma das condições para que a sequência seja convergente é que, na vizinhança da raiz que se pretende encontrar, o declive de $g(x)$ seja menor que 1, que não é o que acontece neste caso:

```
(%i23) subst(x=1.1, diff(tan(x)-1, x));
(%o23)                    4.860280510751842
```

O problema pode ser resolvido quando se consegue inverter a função usada. Ou seja, em vez de se usar $x = \tan(x) - 1$ pode usar-se a equação inversa, $\arctan(x+1) = x$, que implica usar $g(x) = \arctan(x+1)$. No Maxima, a função inversa da tangente é a função `atan`. Com valor inicial $x_0 = 1.1$, os primeiros termos da sequência x_i são:

```
(%i24) g(x) := float (atan(x+1))$
(%i25) xi: 1.1$
(%i26) for i:1 thru 6 do (xi: g(xi), print (xi))$
1.126377116893798
1.131203287160338
1.132075737308623
1.132233108872913
1.132261484138675
1.132266600045208
```

que converge rapidamente para o valor da raiz.

1.5 Método de Newton

Tal como no método de aproximações sucessivas, basta ter um valor inicial para criar uma sequência que se aproxima da raiz. As vantagens em relação ao método de aproximações sucessivas é que não é preciso escrever a equação $f(x) = 0$ na forma $x = g(x)$ e a sequência converge em muitos mais casos. A desvantagem é que é preciso conhecer a derivada, $f'(x)$, da função $f(x)$.

Se $f(x)$ é uma função contínua e derivável, o valor de $f(x_{i+1})$ pode ser calculado, a partir do valor de $f(x_i)$, usando a série de Taylor

$$f(x_{i+1}) = f(x_i) + (x_{i+1} - x_i)\, f'(x_i) + \cdots \tag{1.4}$$

Se $|x_{i+1} - x_i|$ for suficientemente pequeno, os dois primeiros termos da série, apresentados na equação acima, são uma boa aproximação para o valor da série completa. Se o ponto x_{i+1} for uma raiz, então $f(x_{i+1}) = 0$ e a equação anterior conduz à relação de recorrência,

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{1.5}$$

que permite definir uma sequência de valores x_i que deverá aproximar-se do valor da raiz.

Exemplo 1.4
Encontre a raiz de $f(x) = x + 1 - \tan(x)$ mais próxima de $x = 1.1$, com 3 casas decimais, usando o método de Newton.

Resolução. A função $f(x)$ já foi definida no comando (`%i2`) mas como é necessário também definir a sua derivada, vai repetir-se a definição de $f(x)$ para comparar com a forma como $f'(x)$ deve ser definida:

```
(%i27) f(x) := float (x+1-tan(x))$
(%i28) df(x) := float (''(diff (x+1-tan(x), x)));
                                          2
(%o28)                df(x) := float(1 - sec (x))
```

Nas definições de $f(x)$ e $df(x)$ (derivada de $f(x)$), a função `float` não é aplicada imediatamente, mas fica indicada na definição das funções e só será aplicada quando seja dado um valor a x para calcular $f(x)$ ou $df(x)$. Se a função `float` fosse aplicada imediatamente quando $f(x)$ é definida, o resultado seria $x + 1.0 + \tan(x)$ e se mais tarde fosse calculado $f(1)$ o resultado seria $2.0 + \tan(1)$, sem que $\tan(1)$ fosse aproximada por um número de vírgula flutuante.

No entanto, no caso de `diff(x+1+tan(x),x)`, o que se pretende é que a derivada seja calculada imediatamente e o resultado usado para definir a função `df(x)`, em vez de que `diff(x+1+tan(x),x)` seja calculado após ter sido dado um valor para x (uma expressão como `diff(1+1+tan(1),1)` produz um erro). A sintaxe `''`(...) foi usada para forçar a que a expressão dentro dos parêntesis seja calculada e simplificada imediatamente.

Usando o valor inicial $x_0 = 1.1$, os seguintes 6 valores na sequência x_i são:

```
(%i29) xi: 1.1$
(%i30) for i:1 thru 6 do (xi: xi-f(xi)/df(xi), print (xi))$
1.135033812277287
1.13228759380087
1.132267726299738
1.132267725272885
1.132267725272885
1.132267725272885
```

Note-se a rapidez com que o método converge; após apenas 3 interações já se obtêm 4 casas decimais para a raiz e após a quinta iteração já é obtida a solução com o número máximo de casas decimais (15) que é possível obter com o formato de dupla precisão numérica usado pela função `float`.

1.6 Sistemas de equações transcendentes

O método de Newton pode ser generalizado facilmente para resolver um sistema de n variáveis com n equações contínuas e deriváveis. Por exemplo, no caso de duas equações com duas variáveis, $f(x,y)=0$ e $g(x,y)=0$, os primeiros termos nas séries de Taylor para as duas funções são:

$$f(x_{i+1},y_{i+1}) = f(x_i,y_i) + (x_{i+1}-x_i)\left.\frac{\partial f}{\partial x}\right|_{(x_i,y_i)} + (y_{i+1}-y_i)\left.\frac{\partial f}{\partial y}\right|_{(x_i,y_i)} \tag{1.6}$$

$$g(x_{i+1},y_{i+1}) = g(x_i,y_i) + (x_{i+1}-x_i)\left.\frac{\partial g}{\partial x}\right|_{(x_i,y_i)} + (y_{i+1}-y_i)\left.\frac{\partial g}{\partial y}\right|_{(x_i,y_i)} \tag{1.7}$$

Os índices $(x_i,\ y_i)$ indicam que x e y devem ser substituídas por $(x_i,\ y_i)$. Substituindo $f(x_{i+1},y_{i+1})=0$ e $g(x_{i+1},y_{i+1})=0$ e escrevendo o sistema em forma matricial, obtém-se,

$$\mathbf{J}(x_i,y_i)\,(\mathbf{r}_{i+1}-\mathbf{r}_i) = -\mathbf{F}(x_i,y_i) \tag{1.8}$$

onde $\mathbf{J}(x_i,y_i)$ é a **matriz jacobiana**:

$$\mathbf{J}(x,y) = \begin{bmatrix} \dfrac{\partial f}{\partial x} & \dfrac{\partial f}{\partial y} \\[2ex] \dfrac{\partial g}{\partial x} & \dfrac{\partial g}{\partial y} \end{bmatrix} \tag{1.9}$$

calculada em $x=x_i$ e $y=y_i$. As matrizes $\mathbf{r}_{i+1}$, $\mathbf{r}_i$ e $\mathbf{F}(x_i,y_i)$ são:

$$\mathbf{r}_{i+1} = \begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} \qquad \mathbf{r}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \qquad \mathbf{F}(x_i,y_i) = \begin{bmatrix} f(x_i,y_i) \\ g(x_i,y_i) \end{bmatrix} \tag{1.10}$$

A equação (1.8) permite definir a relação de recorrência para x_{i+1} e y_{i+1},

$$\mathbf{r}_{i+1} = \mathbf{r}_i - \mathbf{J}^{-1}(x_i, y_i)\,\mathbf{F}(x_i, y_i) \tag{1.11}$$

onde $\mathbf{J}^{-1}(x,y)$ é a matriz inversa da matriz jacobiana.

Exemplo 1.5
Encontre os pontos de intersecção da circunferência $x^2 + y^2 = 3$ com a hipérbole $y = 1/x$.

Resolução. Os gráficos da circunferência e da hipérbole podem ser traçados com o comando:

```
(%i31) plot2d ([sqrt(3-x^2), -sqrt(3-x^2), 1/x], [x,-3,3],
         [y,-2.25,2.25], [legend,false], [color,blue,blue,red]);
```

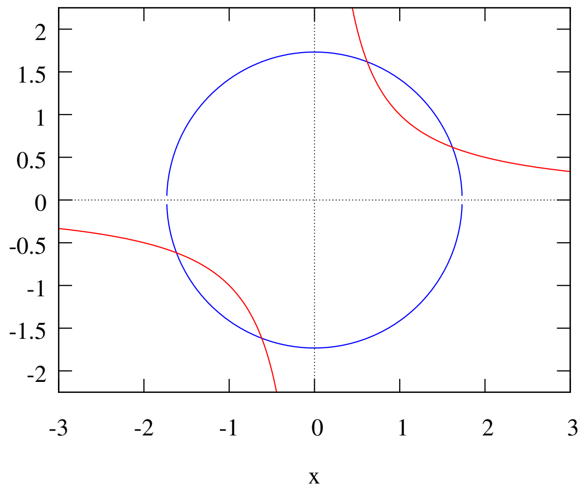

Figura 1.2: Gráficos da circunferência $x^2 + y^2 = 3$ e a hipérbole $y = 1/x$.

O gráfico (figura 1.2) mostra que existem quatro pontos de intersecção, simétricos em relação às retas $y = \pm x$. Basta encontrar um desses quatro pontos, por exemplo, o que está mais próximo do ponto inicial $x_0 = 0.5$, $y_0 = 1$. Escrevendo as duas equações na forma $f(x,y) = 0$ e $g(x,y) = 0$, as duas funções são definidas pelas expressões,

```
(%i32) f: x^2+y^2-3$
(%i33) g: 1-x*y$
```

e a inversa da matriz jacobiana é:

```
(%i34) Jinv: invert (jacobian ([f,g], [x,y]))$
```

Define-se a função vectorial $\mathbf{F}(\mathbf{r}_i)$,

```
(%i35) F(ri) := matrix ([subst ([x=ri[1][1], y=ri[2][1]], f)],
         [subst ([x=ri[1][1], y=ri[2][1]], g)])$
```

A seguir, dá-se o valor inicial para a matriz $\mathbf{r}_i$ e realizam-se algumas iterações:

```
(%i36) ri: matrix ([0.5], [1])$
(%i37) for i:1 thru 6 do
         (ri: ri - subst ([x=ri[1][1], y=ri[2][1]], Jinv).F(ri),
        print (transpose (ri)))$
[ .5833333333333334  1.833333333333333 ]
[ .6089080459770114  1.633908045977012 ]
[ .6178866256040899  1.61819150365287 ]
[ .6180339655308433  1.618034011991991 ]
[ .6180339887498942  1.618033988749896 ]
[ .6180339887498948  1.618033988749895 ]
```

onde o ponto entre `Jinv` e `F(ri)` indica produto matricial.

Tendo em contra a simetria das soluções, as quatro soluções (x, y) são então: (0.6180, 1.6180), (1.6180, 0.6180), (-0.6180, -1.6180) e (-1.6180, -0.6180).