Maxima

Daniel Miranda

6 de Novembro de 2015

Definindo uma lista com os elementos $1, 3, 5, 7, 9$ nessa ordem:

```
( %i1)
[1,3,5,7,9];
(%o1)      [1, 3, 5, 7, 9]
```

Para criar uma lista onde os elementos são definidos por uma expressão, podemos usar o comando makelist.
Para criar a lista onde os elementos são da forma $\frac{1}{n}$ com n de 1 até 8:

```
( %i2)
 makelist(1/n,n,1,8);
(%o2)      [1, 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8]
```

Para criar uma lista com os 10 primeiros impares:

```
( %i3)
 makelist(2*n+1,n,0,10);
(%o3)      [1,3,5,7,9,11,13,15,17,19,21]
```

Para criar a lista onde os elementos são da forma $\frac{n}{n+1}$ com n de 1 até 5:

```
( %i4)
 makelist(n/(n+1),n,1,5);
(%o4)      [1/2,2/3,3/4,4/5,5/6]
```

Manipulando Elementos de uma Lista

Vamos começar criando uma lista:

```
( %i2)
 lista:makelist(n^2,n,1,7);
(%o2)      [1,4,9,16,25,36,49]
```

Os elementos de uma lista são indexados a partir de um. Para retornar o enésimo elemento de uma lista usamos o comando lista[n].

Assim por exemplo, para retornar o primeiro elemento da lista anterior:

```
( %i3)
 lista[1];
(%o3)     1
```

Para retornar o quarto elemento da lista anterior:

```
( %i4)
 lista[4];
(%o4)     16
```

O comando **rest(n)** cria uma nova lista com os *n* primeiros elementos de uma lista removidos:

```
( %i5)
 rest (lista, 3);
(%o5)      [16,25,36,49]
```

O comando **length (lista);** retorna o comprimento de uma lista:

```
( %i6)
 length (lista);
(%o6)    7
```

O comando **append(lista1, lista2)** cria uma nova lista incluindo todos os elementos de lista1 e lista2 seguidos, incluindo os elementos da lista mais longa.

```
( %i7)
 lista1:[a,b,c,d]; lista2:[1,2,3,4,5,6];
```

(%o7) $[a, b, c, d]$

(%o7) $[1, 2, 3, 4, 5, 6]$

```
( %i8)
 append(lista1,lista2);
```

(%o8) $[a, b, c, d, 1, 2, 3, 4, 5, 6]$

os comandos **cons** e **endcons** permitem adicionar novos elementos a uma lista, no início e final de mesma, respectivamente;

```
( %i9)
 cons(X,lista1);
```

(%o9) $[X, a, b, c, d]$

```
( %i10)
 endcons(X,lista1);
```

(%o10) $[a, b, c, d, X]$

O comando **delete** apaga todas as ocorrências da entrada em uma lista

```
( %i11)
delete(a, [a,b,c,a,a,b]);
(%o11)      [b, c, b]
```

É possível realizar operações elementares, tais como a soma, subtração, multiplicação, com os elementos de uma lista:
Vamos calcular a soma dos 20 primeiros pares

```
( %i12)
pares:makelist(2*n,n,1,20);
(%o12)
```

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40]

Para somar os elementos da lista usamos o comando **apply**

```
( %i13)
apply("+", pares)
(%o13)     420
```

Para calcular o produto dos vinte primeiros pares:

```
( %i14)
apply("*", pares)
(%o14)     255108265612582846464000000
```

Para aplicar uma função a cada elemento de uma lista utilizamos o comando **map**:
Para calcular a raiz quadrada dos elementos da lista $[1, 6, 9, 13]$:

```
( %i15)
map(sqrt,[1,6,9,13]);
```

(%o15) $[1, \sqrt{6}, 3, \sqrt{13}]$

Assim por exemplo se $f(x) = x + sin(x)$ e queremos calcular os valores de $f(x)$ para os elementos da lista $[1, 2, 5, 7]$:
Começamos definindo a função $f(x)$ (falaremos mais sobre como definir funções no capítulo ??

```
( %i16)
f(x):=x+sin(x);
```

(%o16) $f(x) := x + sin(x)$

Agora aplicamos $f(x)$ a lista

```
( %i17)
map(f,[1,2,5,7]);
```

(%o17) $[\sin(1) + 1, \sin(2) + 2, \sin(5) + 5, \sin(7) + 7]$

Matrizes

No Maxima define uma matriz do seguinte modo **A: matrix (lista1,lista2,..,listan);**
Vamos definir a matriz

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 3 & -1 & 0 \\ 4 & 2 & 1 \end{pmatrix}$$

```
( %i2)
A:matrix([1,2,3],[3,-1,0],[4,2,1]);
```

(%o2) $\begin{pmatrix} 1 & 2 & 3 \\ 3 & -1 & 0 \\ 4 & 2 & 1 \end{pmatrix}$

Vamos definir outra matriz

```
( %i3)
B: matrix ([1, -1, 2], [2, 1, 5], [0, 1, 3]);
```

$$(\%o3) \quad \begin{pmatrix} 1 & -1 & 2 \\ 2 & 1 & 5 \\ 0 & 1 & 3 \end{pmatrix}$$

Podemos calcular a soma $A + B$ e o produto AB

```
( %i4)
A+B;A.B
```

$$(\%o4) \quad \begin{pmatrix} 2 & 1 & 5 \\ 5 & 0 & 5 \\ 4 & 3 & 4 \end{pmatrix}$$

$$(\%o4) \quad \begin{pmatrix} 1 & -2 & 6 \\ 6 & -1 & 0 \\ 0 & 2 & 3 \end{pmatrix}$$

Cuidado o comando $A * B$ calcula o produto coordenada a coordenada e não o produto de matrizes:

```
( %i5)
C:matrix([2,1],[2,0]);
( %i6)
C.C; C*C;
```

$$(\%o6) \quad \begin{pmatrix} 6 & 2 \\ 4 & 2 \end{pmatrix}$$

$$(\%o6) \quad \begin{pmatrix} 4 & 1 \\ 4 & 0 \end{pmatrix}$$

A Matriz identidade $n \times n$ pode ser facilmente definida usando o comando **ident(n)**

```
( %i7)
id:ident(3);
```

$$(\%o7) \quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Se uma matriz for invertível, sua inversa pode ser calculada através do comando **invert**

```
( %i8)
invert (A);
```

$$(\%o8) \quad \begin{pmatrix} -\frac{1}{23} & \frac{4}{23} & \frac{3}{23} \\ -\frac{3}{23} & -\frac{11}{23} & \frac{9}{23} \\ \frac{10}{23} & \frac{6}{23} & -\frac{7}{23} \end{pmatrix}$$

A transposta de uma matriz pode ser calculada através do comando **transpose**

```
( %i9)
transpose (A);
```

$$(\%o9) \quad \begin{pmatrix} 1 & 3 & 4 \\ 2 & -1 & 2 \\ 3 & 0 & 1 \end{pmatrix}$$

O determinante de uma matriz pode ser calculado atraves do comando **determinant**

```
( %i10)
determinant(A);
```

$$(\%o10) \quad 23$$

A entrada $(2,3)$ da matriz A pode ser obtido através do comando:

```
( %i11)
A[2,3];
(%o11)     0
```

A terceira linha da matriz A pode ser obtida através do comando:

```
( %i12)
row(A, 3);
(%o12)     (4   2   1)
```

A primeira coluna de uma matriz pode ser obtida através do comando:

```
( %i13)
col(A,3);
```

$$(\%o13) \qquad \begin{pmatrix} 3 \\ 0 \\ 1 \end{pmatrix}$$

Uma matriz pode ser escalonada através do comando **triangularize(M)**

(%i14)
```
M:matrix([1,2,3,4],[5,6,7,z],[b,1,2,3]);
```

(%o14)
$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & z \\ b & 1 & 2 & 3 \end{pmatrix}$$

(%i15)
```
triangularize(M);
```

(%o15)
$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & -4 & -8 & z-20 \\ 0 & 0 & -4\,b & (2\,b-1)\,z-24\,b+8 \end{pmatrix}$$

Uma matriz pode ser escalonada com 1 nos pivôs através do comando **echelon(M)**

```
( %i16)
echelon(M);
```

$$(\%o16)\quad \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & -\frac{z-20}{4} \\ 0 & 0 & 1 & -\frac{(2\,b-1)\,z-24\,b+8}{4\,b} \end{pmatrix}$$

Funções

A função $f(x) = x * \sin(x)$ pode ser definida através do comando:

```
( %i1)
f(x):=x*sin(x);
(%o1)      f(x) = x * sin(x)
```

Podemos calcular $f(3)$ através do comando:

```
( %i2)
f(3);
(%o2)      3 sin (3)
```

Podemos calcular $f(0)$ através do comando:

```
( %i3)
f(0);
(%o3)      0
```

(Logo, 0 é uma raiz de $f(x)$)

Podemos calcular $f(a+2)$:

```
( %i4)
f(a+2);
```

(%o4) $(a+2)\sin(a+2)$

De modo análogo podemos definir funções de várias variáveis.

```
( %i5)
g(x,y):=x^2+y^2;
```

(%o5) $g(x,y) := x^2 + y^2;$

Podemos calcular o valor de $g(x, y)$ no ponto $(2, 3)$

```
( %i6)
g(2,3);
(%o6)      13
```

A função definida por partes

$$h(x) = \begin{cases} x + 4 \text{ se } x < 0 \\ 2 \text{ se } x \leq 0 \end{cases}$$

Pode ser definida através do comando;

```
( %i7)
 h(x):=if x<0 then x+4 else 2;
```

(%o7) $h(x) := ifx < 0thenx + 4else2$

```
( %i8)
h(-5);
```

(%o8) -1

```
( %i9)
 meufat(n):=if n=0 then 1 else n*meufat(n-1);
```

$$(\%o9) \qquad meufat\,(n) := if\, n = 0\, then\, 1\, else\, n\, meufat\,(n-1)$$

Vamos calcular *meufat*(n) para 1 até 8

```
( %i10)
 map(meufat,[1,2,3,4,5,6,7,8]);
```

(%o10) $[1, 2, 6, 24, 120, 720, 5040, 40320]$

Gráfico de sin(x) para x entre 0 e π:

```
( %i1)
 wxplot2d (sin(x), [x, 0, 2*%pi]);
(%o1)
```

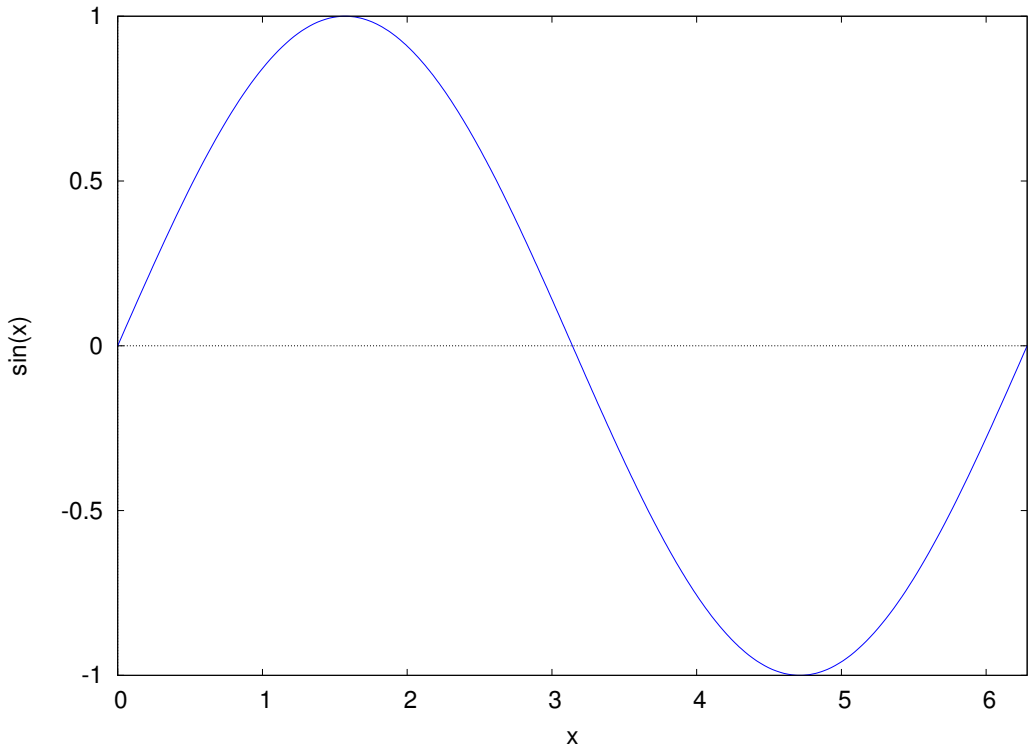
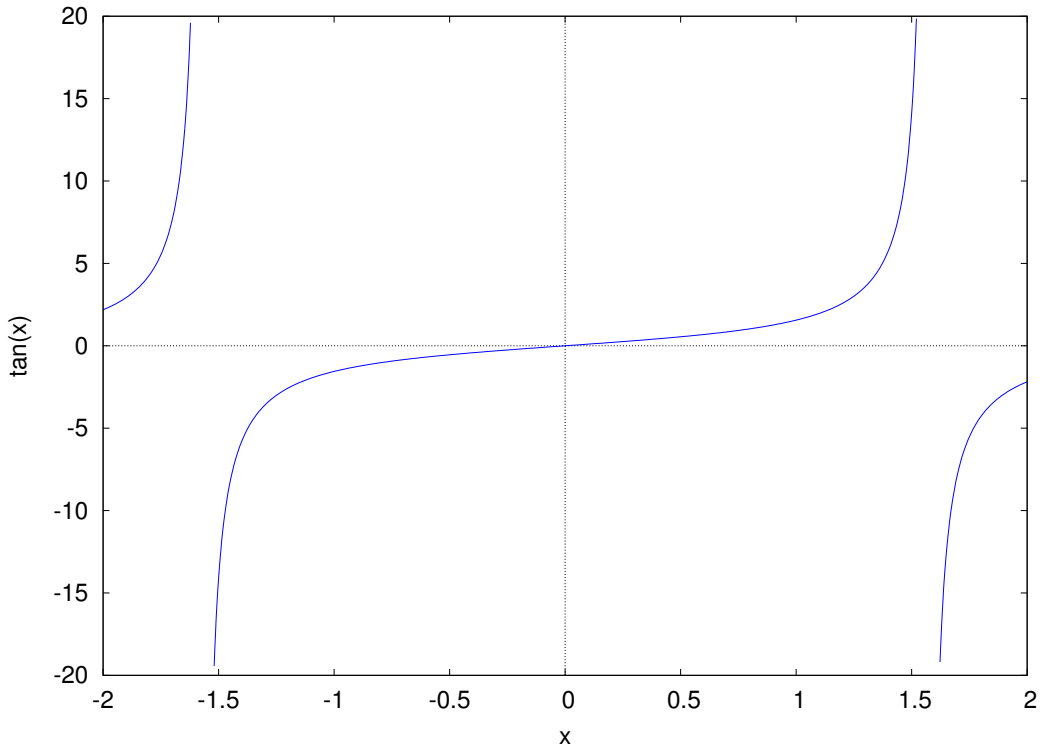
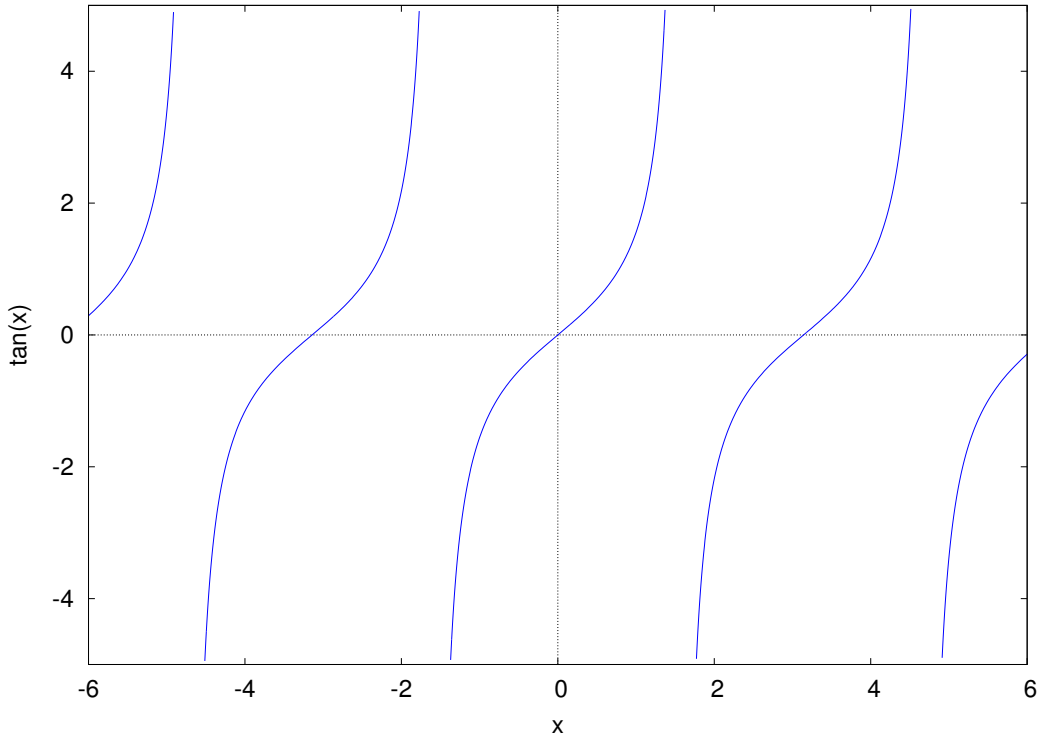


Gráfico de $\sin(x)$ e $\cos(x)$ entre -2π e 2π

```
( %i2)
wxplot2d ([sin(x),cos(x)], [x, -2*%pi, 2*%pi]);
(%o2)
```



```
( %i3)
 wxplot2d([tan(x)], [x,-6,6],[y,-5,5])
(%o3)
```

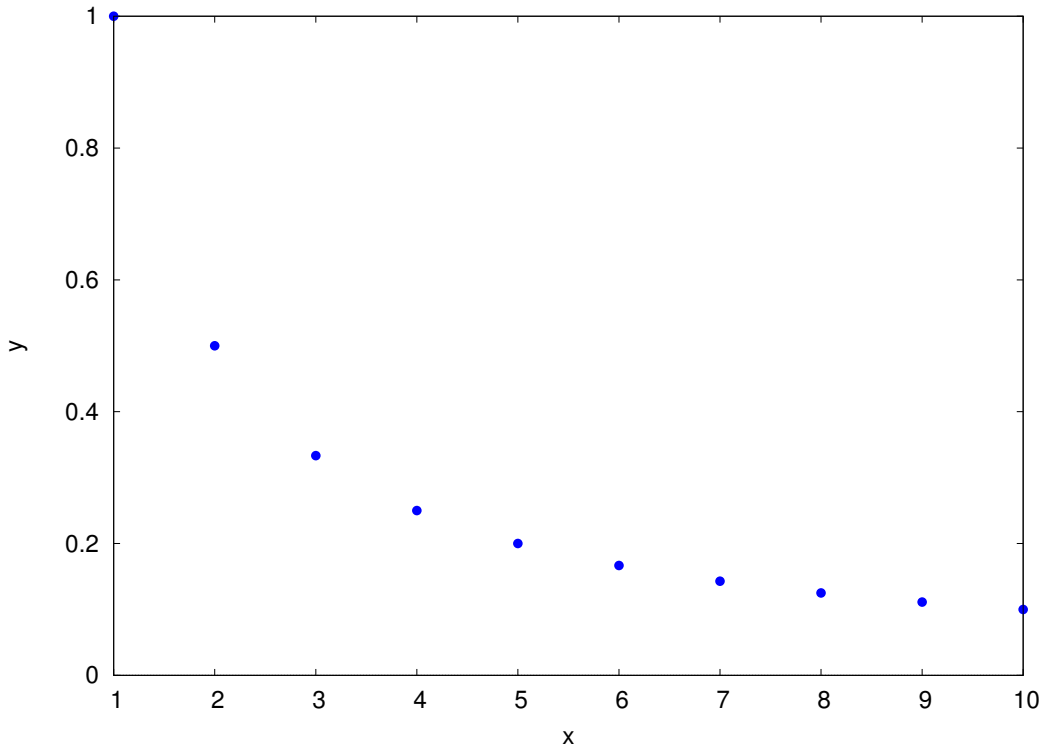


A opção discrete do plot2d permite representar lista de pontos.

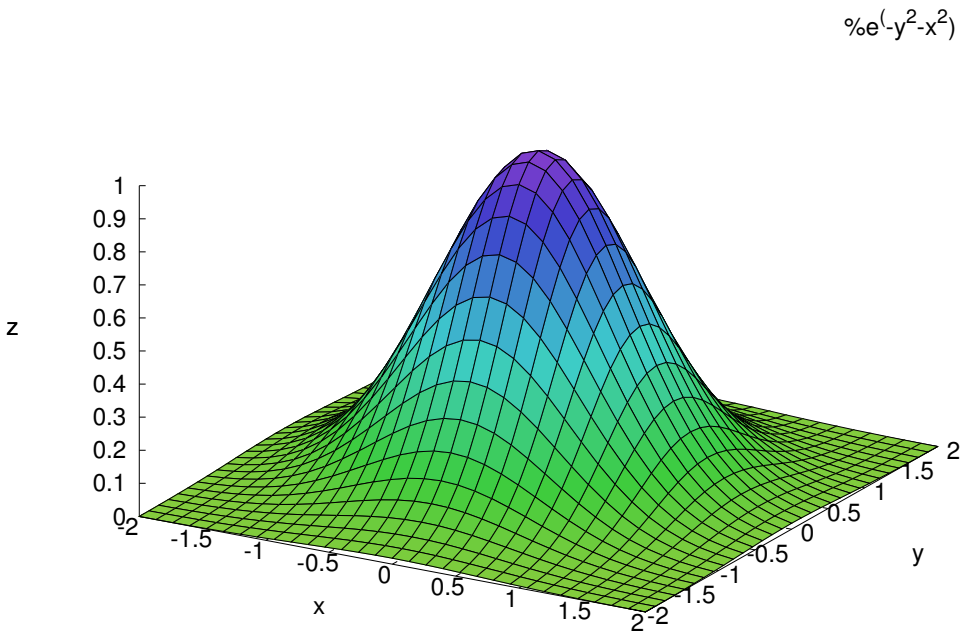
```
( %i4)
sequencia:makelist([n,1/(n)],n,1,10);
(%o4)
[[1,1],[2,1/2],[3,1/3],[4,1/4],[5,1/5],[6,1/6],[7,1/7],[8,1/8],[9,1/9],[10,1/10]]
```

Gráfico

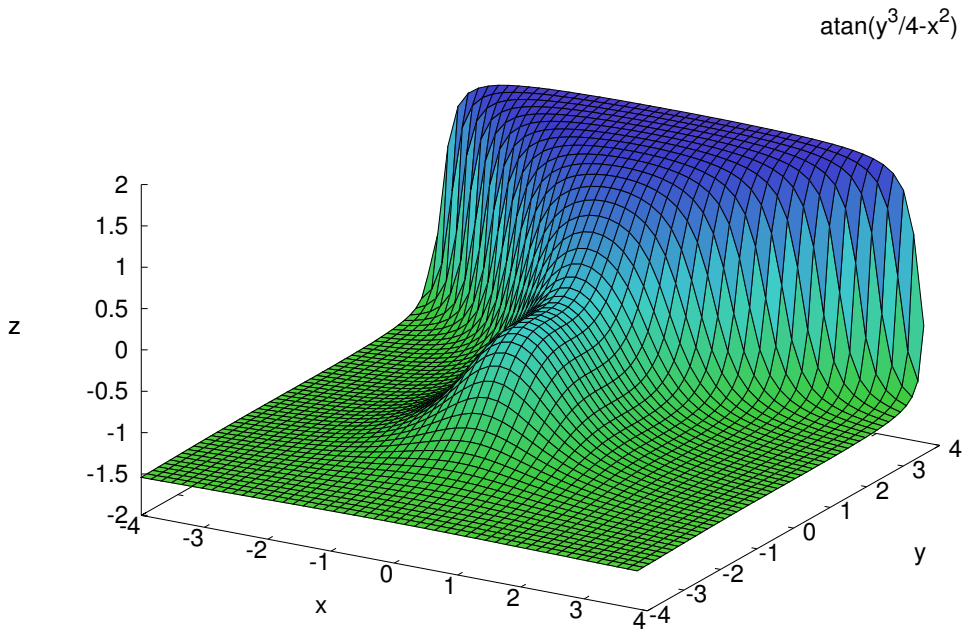
```
( %i5)
wxplot2d([discrete, sequencia], [style, points]);
(%o5)
```



```
( %i6)
plot3d(%e^-(x^2+y^2),[x,-2,2],[y,-2,2]);
(%o6)
```

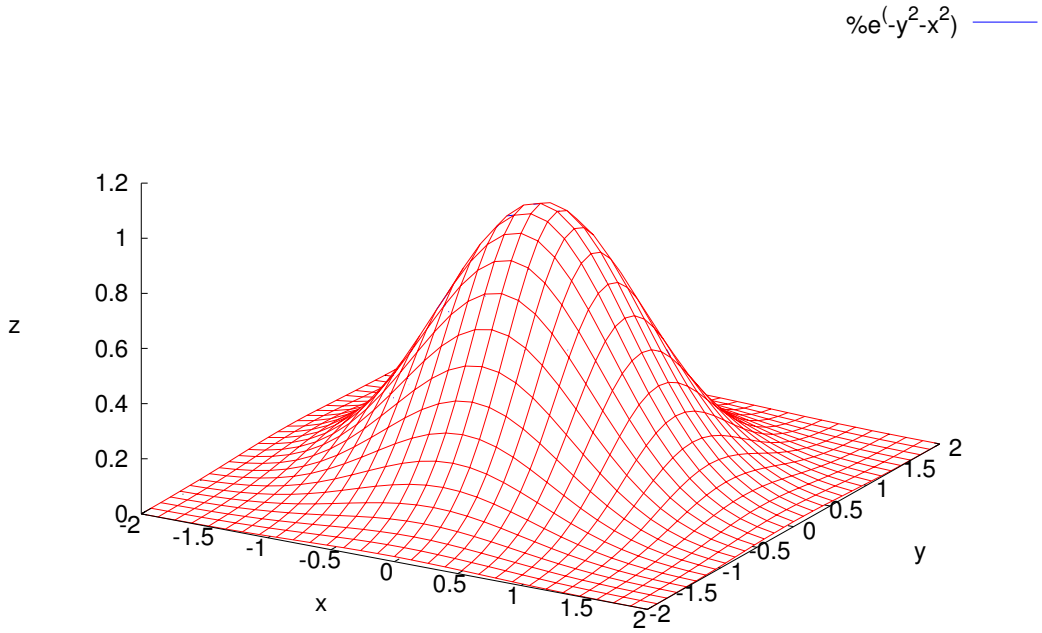


```
( %i7)
plot3d(atan(-x^2+y^3/4),[x,-4,4],[y,-4,4],
[grid,50,50]);
(%o7)
```



Removendo o Mesh

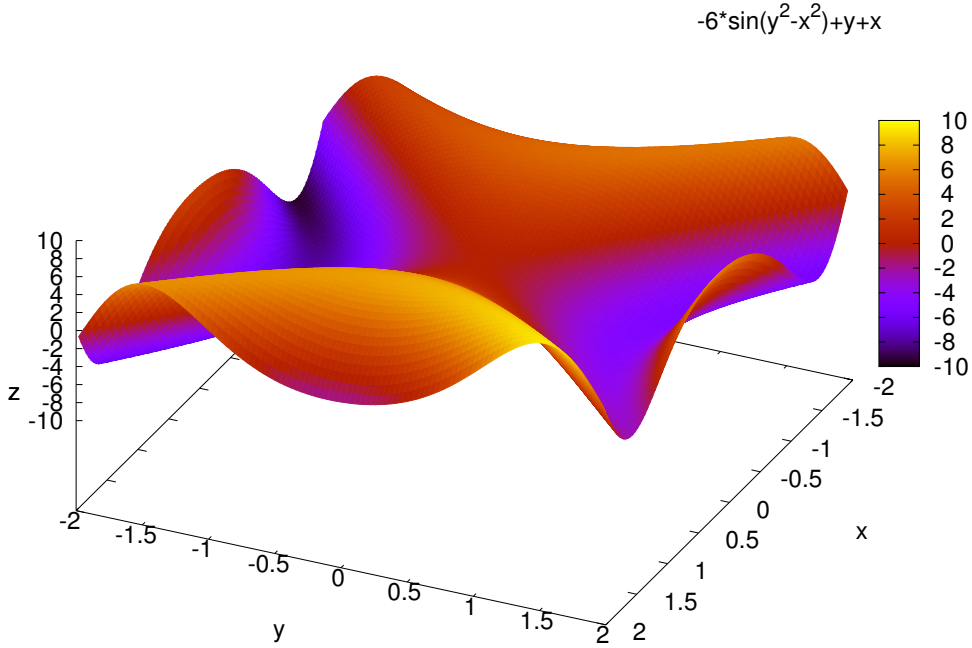
```
( %i8)
plot3d ( %e^(-( x^2+y^2 )), [x, -2, 2], [y, -2, 2],
[z, 0, 1.2],
[palette, false]);
(%o8)
```



Ponto de Vista

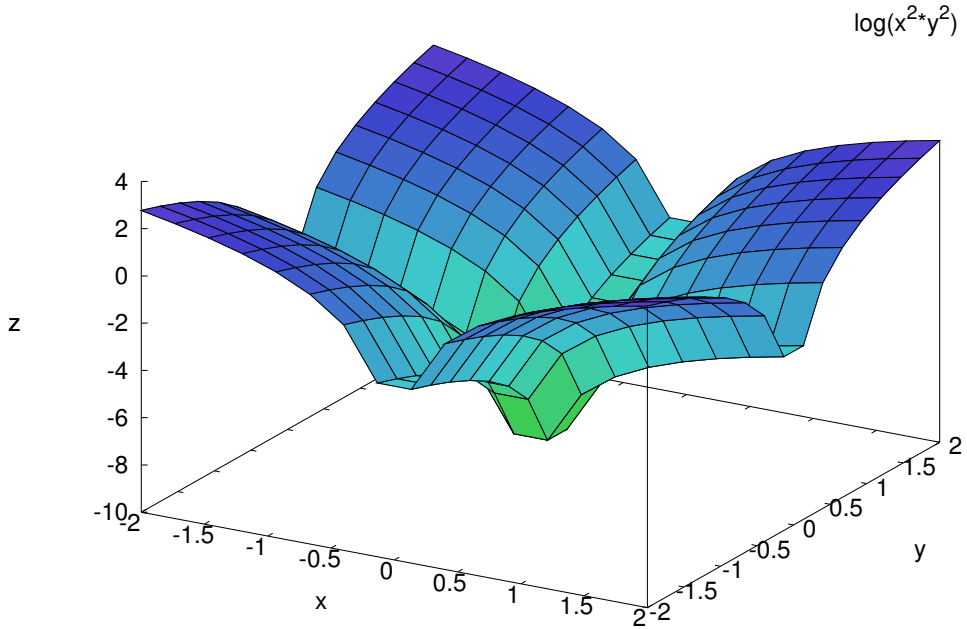
Usando o mouse ou

```
( %i9)
plot3d(6*sin(x^2-y^2)+y^1.3+x, [x, -2, 2],
[y, -2, 2], [grid, 100, 100],
[gnuplot_preamble,"set view 45,115" ]);
(%o9)
```

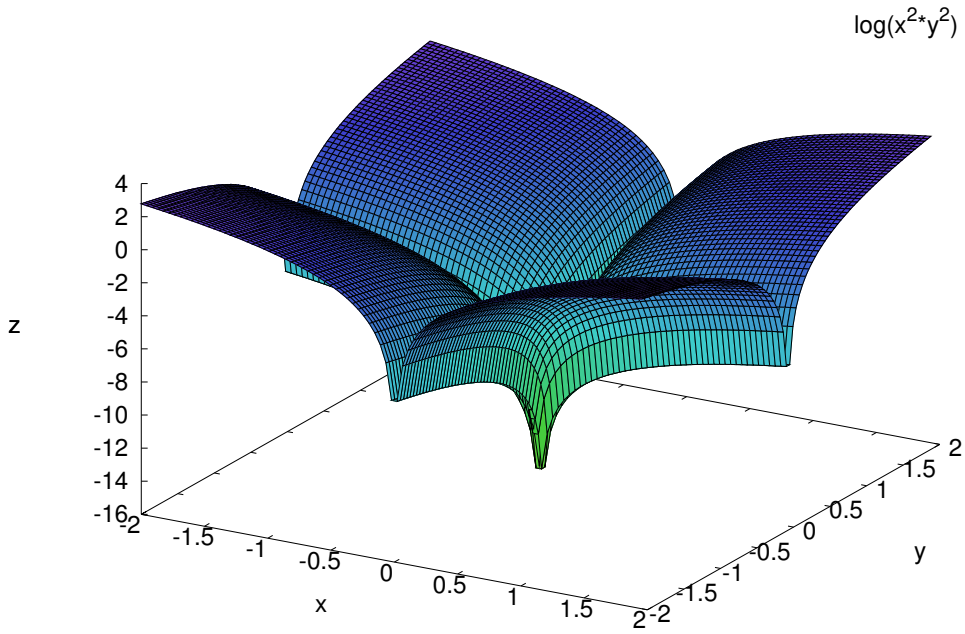


Opções de Grid

```
(%i10)
plot3d(log(x^2*y^2), [x, -2, 2], [y, -2, 2],
[grid, 15, 15])
(%o10)
```

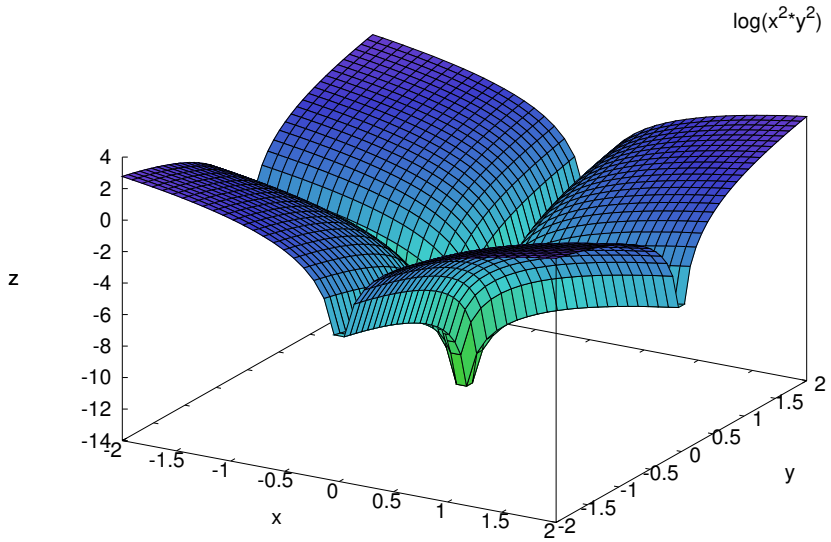


```
( %i11)
plot3d(log(x^2*y^2), [x, -2, 2], [y, -2, 2],
[grid, 91, 91])
(%o11)
```

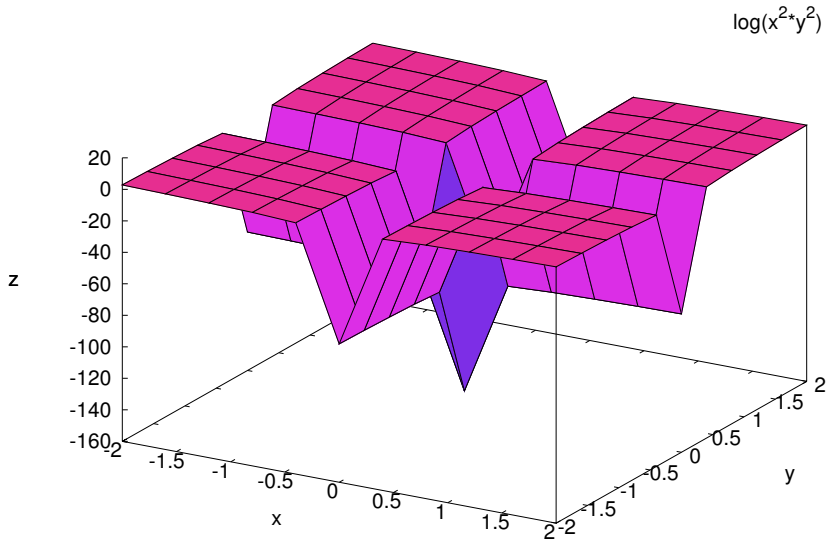


Outras Paletas de Cores

```
(%i12)
plot3d(log(x^2*y^2), [x, -2, 2], [y, -2, 2],
[grid, 29, 29],
 [palette, get_plot_option(palette,2)])
(%o12)
```

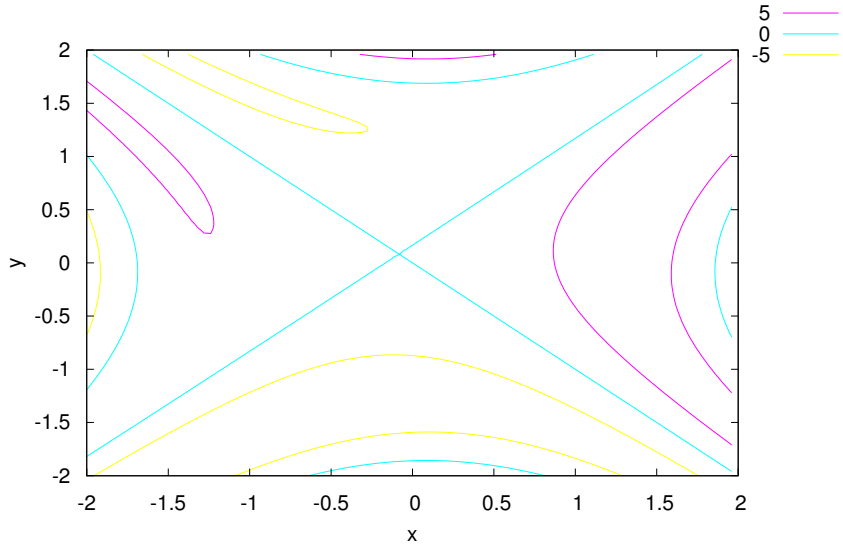


```
( %i13)
plot3d(log(x^2*y^2), [x, -2, 2], [y, -2, 2],
[grid, 29, 29],
      [palette, get_plot_option(palette,4)])
(%o13)
```



Gráficos de Superfícies com Contornos

```
( %i14)
contour_plot( 6*sin(x^2-y^2)+y^1+x, [x, -2, 2],
[y, -2, 2],
[grid, 100, 100])
(%o14)
```



O Maxima pode simplificar ou expandir polinômios usando a função **expand**.
Ao expandir, o Maxima:

- Desenvolverá produtos e potências de somas: por exemplo, $a(b+c)$ será reescrito como $ab+ac$ e $(a+b)^2$ será expandido para $a^2+2ab+b^2$;
- Separará números racionais onde há soma no numerador, reescrevendo como dois números. Por exemplo, $(a+b)/c$ é transformado em $a/c+b/c$;
- Aplicará a propriedade distributiva em multiplicações, dando origem a mais termos. Por exemplo, $a(b+c)$ se torna $ab+ac$.

```
( %i15)
ex: (a+b)*(a+c);
```

(%o15) $(b+a)(c+a)$

```
( %i16)
ratsimp(ex);
```

(%o16) $(b+a)c+ab+a^2$

```
( %i17)
expand(ex);
```

(%o17) $bc+ac+ab+a^2$

A função **ratsimp** tenta simplificar polinômios.

```
( %i18)
p:$x^2(2x-3) +3(2x-4) +8$;
```

(%o18) $x^2(2x-3)+3(2x-4)+8$

```
( %i19)
ratsimp(p);
```

(%o19) $2x^3-3x^2+6x-4$

A função **ratsimp** *não* alterou a forma do polinômio p. Ela apenas encontrou uma versão simplificada do polinômio e a mostrou na tela. O polinômio p continua guardado na mesma forma que antes:

```
( %i20)
p;
```

(%o20) $x^2(2x-3)+3(2x-4)+8$

O Maxima também pode fatorar expressões. Se expandirmos $(a+b)(c+2)^2$, por exemplo,

```
( %i21)
expand((a+b)*(c+2)^2);
```

(%o21) $bc^2 + ac^2 + 4bc + 4ac + 4b + 4a$

Usando **factorsum** na última expressão, teremos de volta a expressão anterior:

```
( %i22)
factorsum(%);
```

(%o22) $(b + a) * (c + 2)^2$

A função **factorsum** é especialmente útil quando temos uma expressão muito grande e queremos verificar rapidamente se ela pode ser fatorada:

```
( %i23)
expressao: b*d^2*e+a*d^2*e+2*b*c*d*e+2*a*c*d*e+b*c^2*e+
a*c^2*e+5*b*d^2+5*a*d^2+10*b*c*d+10*a*c*d+5*b*c^2+5*a*c^2
```

Seria difícil (embora possível) fatorarmos a expressão acima sem a ajuda de um CAS. Usando o Maxima, podemos obter uma forma muito mais simples, fatorada:

```
( %i24)
factorsum(expressao);
```

(%o24) $(b+a)(d+c)^2(e+5)$

O Maxima pode resolver diversos tipos de equação. Como exemplo, encontraremos a solução para $x^2 + 10x + 8 = 0$. Inicialmente digitamos a equação:

```
( %i1)
x^2+10*x+8=0;
```

(%o1) $x^2 + 10x + 8 = 0$

O Maxima simplesmente nos devolveu a equação, sem resolvê-la. A equação é um "objeto matemático" para o Maxima, e ele somente o manipulará se pedirmos explicitamente. A função **solve** encontra soluções para equações e as devolve em uma lista:

```
( %i2)
solve(x^2+10*x+8=0,x);
```

(%o2) $[x = -\sqrt{17} - 5, x = \sqrt{17} - 5]$

Podemos também encontrar os pontos em que duas funções coincidem. No exemplo a seguir daremos ao Maxima dois polinômios, descrevendo uma reta e uma parábola.

```
( %i3)
reta: 2*x+3;
```

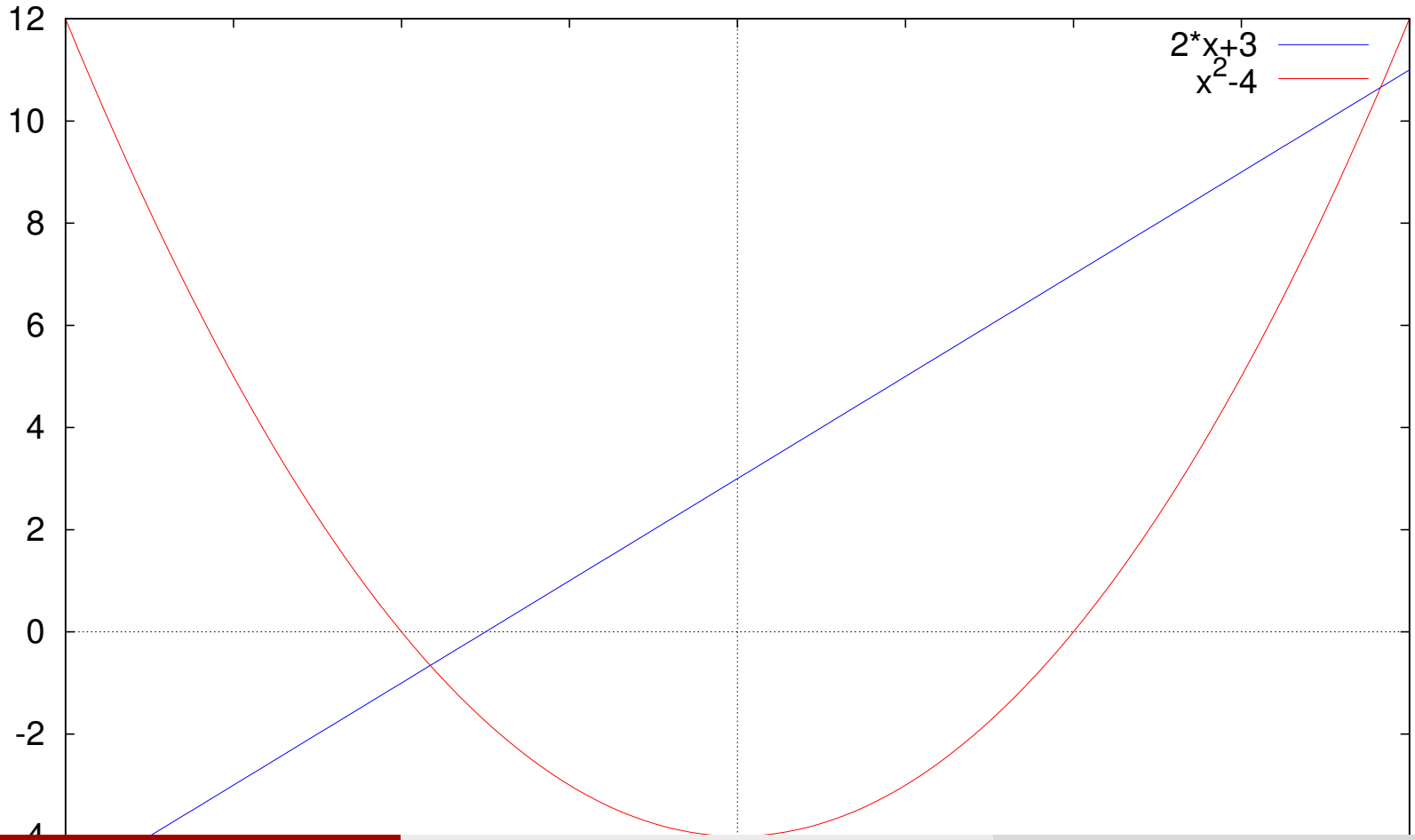
(%o3) $2x + 3$

```
( %i4)
para: x^2-4;
```

(%o4) $x^2 - 4$

Tendo dado nomes aos dois polinômios, podemos então plotá-los e visualizar os pontos onde a reta e a parábola se interceptam:

```
( %i5)
wxplot2d([reta,para],[x,-4,4]);
(%o5)
```



Também podemos verificar os zeros de ambos:

```
( %i7)
solve(reta=0);
```

(%o7) $[x = -\frac{3}{2}]$

```
( %i8)
solve(para=0);
```

(%o8) $[x = -2, x = 2]$

Além de equações, a função **solve** também resolve sistemas de equações lineares. Basta que passemos como primeiro argumento o sistema de equações como uma lista:

```
( %i9)
solve([x+y=2, 2*x-y=4]);
```

(%o9) $[[y = 0, x = 2]]$

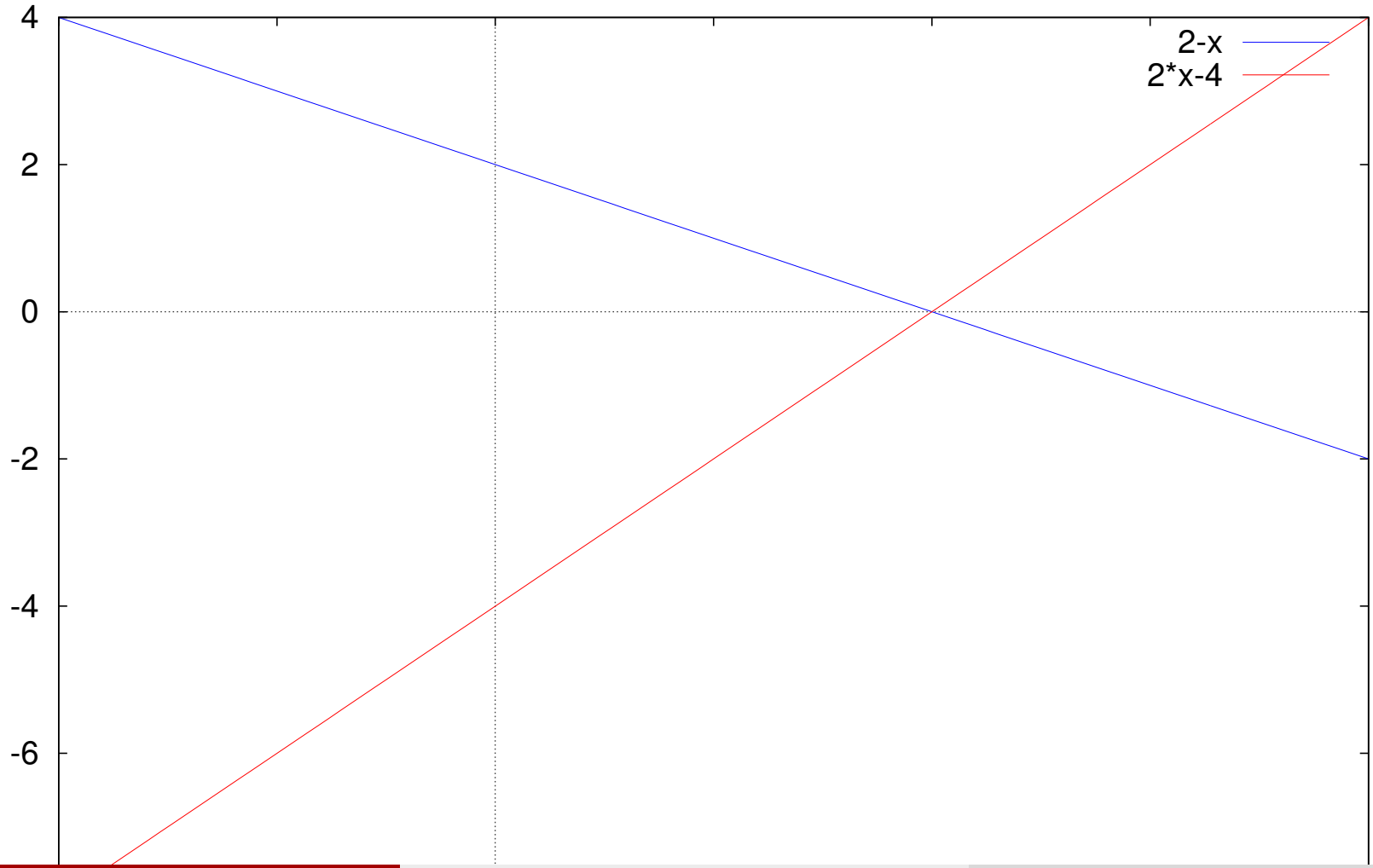
O sistema 2×2 que acabamos de resolver é composto de duas equações afim. Podemos plotá-las e visualizar o ponto onde se interceptam. Para isto devemos reescrever ambas na forma $y = f(x)$ – e o próprio Maxima o fará se pedirmos que resolva cada linha do sistema para y:

```
( %i10)
solve(x+y=2,y);
(%o10)      [y = 2 - x]

( %i11)
solve(2*x-y=4,y);
(%o11)      [y = 2x - 4]
```

Finalmente, plotamos as duas funções:

```
( %i12)
wxplot2d([2-x, 2*x-4],[x,-2,4]);
(%o12)
```



Para que o Maxima resolva inequações, é necessário carregar o módulo **fourier_elim** (o módulo tem este nome porque usa o *processo de eliminação de Fourier-Motzkin* para resolver sistemas de inequações lineares):

```
( %i1)
load(fourier_elim);
```

O Maxima então pode calcular as soluções para inequações. Primeiro uma inequação simples ($3x - 10 - x > 0$):

```
( %i2)
fourier_elim([3*x-10-x>0],[x]);
(%o2)      [5 < x]
```

Uma inequação com módulo ($|3x - 10| - x > 0$):

```
( %i3)
fourier_elim([abs(3*x-10)-x>0],[x]);
(%o3)      [5 < x] or [x < 5/2]
```

Uma inequação com dois módulos, um dentro do outro:

$$| x - |4 - x| | < 2, \tag{1}$$

Basta que passemos à função **fourier_elim** a inequação e a variável:

```
( %i4)
fourier_elim(abs(x - abs(4-x)) < 2,[x]);
(%o4)      [1 < x, x < 3]
```

O segundo argumento para **fourier_elim** foi $[x]$, uma lista contendo x. Esta lista contém as variáveis que o Maxima levará em consideração ao resolver as inequações. Para ilustrar isto, usaremos uma inequação com duas variáveis, $xy > 5x$:

```
( %i5)
fourier_elim(x * y > 5*x,[x]);
```

(%o5) $[0 < x, y - 5 > 0] or [x < 0, 5 - y > 0]$

A resposta do Maxima mostra diversas inequações, mas em todas as que envolvem x ele aparece isolado. Se passarmos $[y]$ para **fourier_elim**, o Maxima resolverá a inequação para y:

```
( %i6)
fourier_elim(x * y > 5*x,[y]);
```

(%o6) $[5 < y, x > 0] or [y < 5, -x > 0]$

Para calcularmos limites usamos o comando limit. **limit(função, variavel ponto)**
Assim por exemplo para calcularmos o limite:

$$\lim_{x \to 0} \frac{\sin(5x)}{sin(3x)}$$

```
( %i1)
limit(sin(5*x)/sin(3*x),x,0);
(%o1)     5/3
```

```
( %i1)
diff(sin(x),x);
(%o1)      cos(x)
```

```
( %i1)
integrate(sin(x/4),x);
```

(%o1) $-4\cos\left(\frac{x}{4}\right)$

```
( %i2)
integrate(x/(x^3+1),x);
```

(%o2) $\frac{\log\left(x^2-x+1\right)}{6}+\frac{\operatorname{atan}\left(\frac{2x-1}{\sqrt{3}}\right)}{\sqrt{3}}-\frac{\log(x+1)}{3}$

```
( %i3)
integrate(sin(x/4),x,1,2);
```

(%o3) $4\left(\cos\left(\frac{1}{4}\right)-\cos\left(\frac{1}{2}\right)\right)$