Maxima快速参考手册*

版本 1.1

马欢†

May 24, 2011

目录

1 基本介绍 **4**
1.1 一点历史 4
1.2 启动和退出 4
1.3 在线帮助 5
1.4 数据类型 6

2 初等数学 **8**
2.1 算术运算 8
2.2 常用初等函数 8
2.3 预定义常数 8
2.4 自定义函数 8
2.5 求和与求积 8
2.6 代数运算与化简 9
2.6.1 多项式展开 9
2.6.2 因式分解 10
2.6.3 等量代换 10
2.6.4 分式展开 10
2.6.5 分式化简 10
2.6.6 对数、指数及根式化简 11
2.6.7 条件假设 11
2.7 三角函数变换 12

*Maxima快速参考手册 by Huan Ma is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.

†Copyright ©2010–2011 Huan Ma. 欢迎反馈：yusufma77@yahoo.com

2.7.1 常用变换 12
2.7.2 控制变量 13
2.8 解方程 13
2.8.1 单个方程 13
2.8.2 方程组 13
2.8.3 数值解 14
2.9 复数 15
2.9.1 实部与虚部 15
2.9.2 复共轭 15
2.9.3 复数模和辐角 15
2.9.4 直角形式和极坐标形式 16

3 矩阵 **17**
3.1 矩阵输入 17
3.1.1 交互式输入 17
3.1.2 以列表形式输入 17
3.1.3 以函数形式输入 17
3.1.4 对角矩阵 18
3.1.5 单一非零元素矩阵 18
3.2 矩阵运算 18
3.2.1 加减乘除 18
3.2.2 幂运算 19
3.2.3 矩阵乘法 19
3.3 行（列）操作 19
3.3.1 提取行（列） 19
3.3.2 增加行（列） 20
3.3.3 子矩阵 20
3.4 线性代数 21
3.4.1 行列式 21
3.4.2 矩阵转置 21
3.4.3 矩阵的逆 21
3.4.4 矩阵的秩 21
3.4.5 高斯消元 22
3.4.6 本征值和本征向量 22

4 微积分 **23**
4.1 微分 23
4.2 积分 23
4.2.1 不定积分 23
4.2.2 定积分 23

4.2.3 数值积分 24
4.3 泰勒展开 25
4.4 拉普拉斯变换 25
4.5 留数 26

5 微分方程 **27**
5.1 一阶或二阶常微分方程通解 27
5.2 初值问题 27
5.3 边值问题 27
5.4 一阶线性微分方程组 28

6 特殊函数 **29**

7 作图 **30**
7.1 二维作图 30
7.1.1 一般函数作图 30
7.1.2 对数坐标图 31
7.1.3 参数方程作图 31
7.1.4 数据作图 31
7.1.5 存为图片文件 33
7.2 三维作图 33

8 图形界面 **35**
8.1 wxMaxima 35
8.2 xmaxima 36
8.3 TeXmacs 37
8.4 Emacs+imaxima 37

9 Maxima编程 **39**
9.1 do循环 39
9.2 if条件句 40
9.3 block程序块 41
9.4 read读取 41
9.5 结果输出 41
9.5.1 二维表示 41
9.5.2 输出到文件 41
9.6 随机数 42
9.7 batch执行程序 42
9.8 tex输出 42

索引 **43**

1 基本介绍

1.1 一点历史

Maxima是一个计算机代数运算系统，它是由MIT于1968至1982年间开发的Macsyma系统演变而来的。MIT于1982年将Macsyma源代码交与美国能源部（DOE），这部分代码后来被称作DOE Macsyma。DOE Macsyma的一个版本从1982年起由德克萨斯大学的William F. Schelter教授维护，直到Schelter教授于2001年去逝。在1998年，Schelter教授从能源部获得了授权，将DOE Macsyma源代码以GNU Public License版权协议发布。在2000年，他在SourceForge开始了Maxima项目，继续维护开发DOE Macsyma，自此该软件被称作Maxima。

Maxima的安装并不复杂，本文不作介绍。基本上所有主要Linux发行版都包含Maxima软件，而Windows用户可以去Maxima网站下载安装软件。本文中的Maxima代码均在Linux上完成测试，Windows下应该大同小异。

1.2 启动和退出

打开一个终端窗口，输入命令：

```
maxima
```

Maxima将会启动，你会看到版本信息和提示符：

```
Maxima 5.19.2 http://maxima.sourceforge.net
Using Lisp CLISP 2.47 (2008-10-23)
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
The function bug_report() provides bug reporting information.
(%i1)
```

其中`(%i1)`是输入提示符，表示第1个输入命令。相应的，在输出的结果之前，你会看到类似于`(%o1)`的行标，这表示第1个输出结果。

如果要退出，输入：

```
(%i1) quit();
```

注意：

- 命令*quit*之后有个圆括号，这是因为*quit*也是一个函数，只不过是一个没有包含任何自变量的函数。
- 命令之后有个分号。*Maxima*和*C*语言类似，都使用分号作为语句的结尾。
- *Maxima*是区分大小写的。*Maxima*预定义的函数都是小写形式。

1.3 在线帮助

在介绍任何其他命令之前，我们要先学会如何使用帮助。使用问号?（精确查找）或者双问号??（模糊查找）可以看到某个命令的帮助信息。

```
(%i1) ? real
  No exact match found for topic `real'.
  Try `?? real' (inexact match) instead.

(%o1)                                false
(%i2) ?? real

 0: absolute_real_time  (Functions and Variables for Runtime Environment)
 1: elapsed_real_time  (Functions and Variables for Runtime Environment)
 2: Real infinity  (Functions and Variables for Constants)
 3: realonly  (Functions and Variables for Equations)
 4: realpart  (Functions and Variables for Expressions)
 5: realroots  (Functions and Variables for Equations)
Enter space-separated numbers, `all' or `none': 4

 -- Function: realpart (<expr>)
     Returns the real part of <expr>. `realpart' and `imagpart' will
     work on expressions involving trigonometic and hyperbolic
     functions, as well as square root, logarithm, and exponentiation.


(%o2)                                true
```

另外，使用`example`命令可以查看某个命令的使用范例。

```
(%i1) example(realpart);
(%i2) %e^(%i*alpha)+(u+%i*v)/(%i*e+f)
                              %i v + u      %i alpha
(%o2)                         -------- + %e
                              f + %i e
(%i3) realpart(%)
                             e v + f u
(%o3)                        ---------- + cos(alpha)
                               2     2
                              f  + e
(%o3)                                done
```

1.4 数据类型

Maxima有四种基本的数据类型：整数，有理数（分数），普通浮点数（小数），大浮点数（任意精度的小数）。其他的数据类型是这几种类型的组合，例如复数可以看作是一对数的组合。Maxima是一个精确运算的代数系统。也就是说，如果你给出的数字是整数，运算结果不会自动转换为小数（浮点数），而是给出分数形式。如果运算包含开根号，Maxima会在结果中保留根号，而不是给出一个计算出的小数值。但是，有时候你可能想要最终的结果为小数，这时候可以打开`numer`开关给出小数结果，或者用`float`或`bfloat`将结果分别转化为浮点数或者大浮点数。大浮点数的精度由变量`fpprec`控制。例子：

```
(%i1) sqrt(5) + 2/3;
                                          2
(%o1)                              sqrt(5) + -
                                          3
(%i2) float(%);
(%o2)                            2.902734644166456
(%i3) bfloat(%o1);
(%o3)                           2.902734644166456b0
(%i4) sqrt(5) + 2/3, numer;
(%o4)                            2.902734644166456
(%i5) fpprec:50$
(%i6) bfloat(%o1);
(%o6)        2.9027346441664563630758403353979429021072850262782b0
```

其中`%`表示最后一次运算的结果。另外，输出行标也可以用作表示以前的输出结果。

注意：

- 第5行输入里的冒号表示赋值。在*Maxima*里等号=用于判断相等关系，不用来赋值。
- 第5行输入并没有使用分号作为语句结尾，而是使用了$作为结尾。这是另一种表示语句结尾的方式，这时不会有输出行打印。当我们并不那么关心运算结果的时候（例如赋值语句），可以使用这种结尾方式。

浮点数在运算过程中会“感染”其他数据类型。例如，如果运算对象包含大浮点数，结果也为大浮点数；如果运算对象没有大浮点数但是包含普通浮点数，结果也为普通浮点数。例如：

```
(%i1) 1/2 + 3/5;
                                        11
(%o1)                                   --
                                        10
(%i2) 0.5 + 3/5;
(%o2)                                   1.1
(%i3) 5b-1 + 3/5;
(%o3)                                 1.1b0
```

这里的5b-1中的b表示大浮点数。

对于一组数，我们可以用由方括号包围的列表表示：

```
(%i1) [2, 3, 4] + [7, 8, 9];
(%o1)                                    [9, 11, 13]
```

2 初等数学

2.1 算术运算

符号`+ - * / ^`分别表示加，减，乘，除和幂运算。另外，`**`也可以用来表示幂运算。阶乘运算由算符`!`表示。

2.2 常用初等函数

这里只列出名字，函数名已经解释了这些函数的用法。

sqrt()	abs()	max()	min()	sign()	exp()
sin()	cos()	tan()	cot()	sec	csc()
sinh()	cosh()	tanh()	coth()	sech	csch()
asin()	acos()	atan()	acot()	asec()	acsc()
asinh()	acosh()	atanh	acoth()	asech()	acsch()
log()					

2.3 预定义常数

Maxima有一些预先定义好的常数。最常用的有如下几个:

符号	名称
`%e`	自然对数的底（e）
`%i`	虚数单位（i）
`%pi`	圆周率（π）
`inf`	实数正无穷
`minf`	实数负无穷
`infinity`	复数无穷

2.4 自定义函数

Maxima使用`:=`算符来定义函数。

```
(%i1) f(x, y) := sin(x) - cos(2*y);
(%o1)                    f(x, y) := sin(x) - cos(2 y)
(%i2) f(%pi/4, %pi/8);
(%o2)                                  0
```

2.5 求和与求积

求和符号$\sum_{i=m}^{n} x_i$在Maxima里由函数`sum(expr, i, m, n)`表示，其中`expr`为一表达式。

```
(%i1) sum (i^2, i, 1, 10);
(%o1)                                   385
(%i2) sum (1/2^i, i, 1, n);
                                      n
                                    ====
                                    \      1
(%o2)                                >     --
                                    /      i
                                    ====   2
                                    i = 1
(%i3) sum (1/2^i, i, 1, n), simpsum;
                                      - n - 1    1
(%o3)                           - 2 (2          - -)
                                                   2
(%i4) sum (1/2^i, i, 1, inf), simpsum;
(%o4)                                    1
```

由上面的例子看出，求和函数的输出只是给出形式上的答案，不会自动对结果进行化简。如果要让Maxima尝试化简，需要打开`simpsum`开关。

相应的，求积符号$\prod_{i=m}^{n} x_i$在Maxima里由函数`product(expr, i, m, n)`表示。让求积函数尝试化简的开关是`simpproduct`。

```
(%i1) product (i, i, 1, n);
                                      n
                                   /===\
                                    ! !
(%o1)                               ! !   i
                                    ! !
                                   i = 1
(%i2) product (i, i, 1, n), simpproduct;
(%o2)                                   n!
```

2.6 代数运算与化简

2.6.1 多项式展开

```
(%i1) (x + 2*y)^3;
                                          3
(%o1)                              (2 y + x)
(%i2) expand(%);
                          3          2        2        3
(%o2)                   8 y  + 12 x y  + 6 x  y + x
```

2.6.2 因式分解

```
(%i1) 4*x^3 - 3*x*y^2 - y^3;
```

(%o1) $$-y^3-3\,x\,y^2+4\,x^3$$

```
(%i2) factor(%);
```

(%o2) $$-(y-x)\,(y+2\,x)^2$$

2.6.3 等量代换

```
(%i1) x^2 + x + 1/x, x = 1/y;
```

(%o1) $$y+\frac{1}{y}+\frac{1}{y^2}$$

```
(%i2) subst(1/a, y, %);
```

(%o2) $$a^2+a+\frac{1}{a}$$

2.6.4 分式展开

```
(%i1) 1/(x - 1) + (x - 1)/(x + 1);
```

(%o1) $$\frac{x-1}{x+1}+\frac{1}{x-1}$$

```
(%i2) expand(%);
```

(%o2) $$\frac{x}{x+1}-\frac{1}{x+1}+\frac{1}{x-1}$$

```
(%i3) ratexpand(%o1);
```

(%o3) $$\frac{x^2}{x^2-1}-\frac{x}{x^2-1}+\frac{2}{x^2-1}$$

从上面的例子可以看出来，命令`ratexpand`和`expand`的区别在于，`ratexpand`会进行通分操作。

2.6.5 分式化简

```
(%i1) ((x - 1)^(3/2) - (x + 1)*sqrt(x - 1))/sqrt((x - 1)*(x + 1));
```

(%o1)
$$\frac{(x-1)^{3/2} - \mathrm{sqrt}(x-1)\,(x+1)}{\mathrm{sqrt}((x-1)\,(x+1))}$$

```
(%i2) ratsimp(%);
```

(%o2)
$$-\frac{2\,\mathrm{sqrt}(x-1)}{\mathrm{sqrt}(x^2-1)}$$

2.6.6 对数、指数及根式化简

```
(%i1) (log(x+x^2)-log(x))^a/log(1+x)^(a/2);
```

(%o1)
$$\frac{(\log(x^2+x)-\log(x))^a}{\log(x+1)^{a/2}}$$

```
(%i2) radcan(%);
```

(%o2)
$$\log(x+1)^{a/2}$$

```
(%i3) (%e^x-1)/(1+%e^(x/2));
```

(%o3)
$$\frac{\%e^x-1}{\%e^{x/2}+1}$$

```
(%i4) radcan(%);
```

(%o4)
$$\%e^{x/2}-1$$

2.6.7 条件假设

有时化简结果依赖于某些条件，这时候可以用`assume`来预定假设条件。

```
(%i1) sqrt (x^2);
```

(%o1) $\mathrm{abs}(x)$

```
(%i2) assume (x<0);
```

(%o2) $[x < 0]$

```
(%i3) sqrt (x^2);
```

(%o3) $-x$

使用`facts()`命令可以查看当前假设，`forget`可以取消某项假设条件。

另外，使用`let`和`letsimp`组合可以更为灵活的设置简化条件。

```
(%i1) f(x)^2 / (1/g(x)^2 - 1) + g(x)^2 / (1/f(x)^2 - 1);
                                   2            2
                                 g (x)        f (x)
(%o1)                          ---------- + ----------
                                 1            1
                               ----- - 1    ----- - 1
                                 2            2
                               f (x)        g (x)
(%i2) let (g(x)^2, 1 - f(x)^2);
                                   2              2
(%o2)                            g (x) --> 1 - f (x)
(%i3) letsimp (%o1);
(%o3)                                  1
```

2.7 三角函数变换

2.7.1 常用变换

常用的三角函数变换有下面几种:

`trigexpand`	利用和差化积公式展开
`trigreduce`	利用积化和差公式变成`sin`或`cos`的和
`trigsimp`	利用$\sin^2(x)+\cos^2(x)=1$等公式简化
`trigrat`	简化分数形式，分子分母为`sin`和`cos`的线性函数

```
(%i1) sin(2*x)/cos(x) + cos(2*x);
                                 sin(2 x)
(%o1)                            -------- + cos(2 x)
                                  cos(x)
(%i2) trigexpand(%);
                                 2                    2
(%o2)                     - sin (x) + 2 sin(x) + cos (x)
(%i3) trigreduce(%);
                      cos(2 x) + 1    cos(2 x)                1
(%o3)                 ------------ + -------- + 2 sin(x) - -
                          2             2                    2
(%i4) trigsimp(%);
(%o4)                            cos(2 x) + 2 sin(x)
(%i5) trigrat(%o1);
(%o5)                            cos(2 x) + 2 sin(x)
```

2.7.2 控制变量

`triginverses`变量用于控制反三角函数的简化，有三个可能的值：

`all`	`arcfun(fun(x))`和`fun(arcfun(x))`都会简化为`x`
`true`	只有`fun(arcfun(x))`会简化为`x`
`false`	`arcfun(fun(x))`和`fun(arcfun(x))`都不会简化为`x`

2.8 解方程

2.8.1 单个方程

使用`solve (eqn, x)`命令解方程。`eqn`是待解方程，`x`是待解变量。如果`eqn`不含等号而只是一个表达式，则Maxima默认为`eqn = 0`。如果方程只包含一个变量，`x`也可以省去。

```
(%i1) solve (a*x^2 + b*x + c, x);
                                 2                          2
                        sqrt(b  - 4 a c) + b       sqrt(b  - 4 a c) - b
(%o1)          [x = - ---------------------, x = ---------------------]
                              2 a                        2 a
(%i2) solve (exp(x^2 + x)  = 2);
                      sqrt(4 log(2) + 1) + 1       sqrt(4 log(2) + 1) - 1
(%o2)        [x = - ------------------------, x = ------------------------]
                              2                            2
```

2.8.2 方程组

`solve`命令也可以用来解方程组，方程组和变量都用列表来表示。

```
(%i1) eq1: b + 2*c = 0;
(%o1)                                  2 c + b = 0
(%i2) eq2: 2*a*b - c*b = 5;
(%o2)                               2 a b - b c = 5
(%i3) solve ([eq1, eq2], [a, b]);
                                        2
                                   2 c  - 5
(%o3)                          [[a = --------, b = - 2 c]]
                                     4 c
(%i4) solve ([(x-1)^2 + y^2 = 4, x*y=1], [x,y]);
(%o4) [[x = .5153595952294904, y = 1.94039270687237],
[x = .3167481152931008 %i - .7434214059810598,
y = - .4850624940594348 %i - 1.13846246879373],
[x = - .3167481152931008 %i - .7434214059810598,
y = .4850624940594348 %i - 1.13846246879373],
[x = 2.971483220309511, y = .3365322722218971]]
```

由上面的例子看出，`solve`会根据具体情况调用不同的算法，在无法获得准确解的情况下，也可能给出近似的数值解。

2.8.3 数值解

多项式的数值解 函数`allroots`用来发现多项式所有近似数值解，而`realroots`只返回实数解的近似值。

```
(%i1) (2*x - 1)^3 - 3*x^2 + 4 = 0;
                                    3       2
(%o1)                        (2 x - 1)  - 3 x  + 4 = 0
(%i2) allroots (%);
(%o2) [x = - .2780531233475246, x = .4356072860855172 %i + 1.076526561673762,
                                  x = 1.076526561673762 - .4356072860855172 %i]
(%i3) realroots (%o1);
                                        9329915
(%o3)                              [x = - --------]
                                        33554432
```

注意：

- 虽然*realroots*返回的结果是分数形式，但是它也是近似值。

线性插值法 对于连续函数`f`，可以用`find_root(f, x, a, b)`在区间`[a, b]`上寻找`f = 0`的数值解。

注意：

- 函数f在a和b的取值应当符号相反。
- 当区间$[a, b]$内有超过一个解时，*find_root*只能返回其中一个。

```
(%i1) f: x^2 + sin(5*x-2) - 1;
                                                2
(%o1)                          sin(5 x - 2) + x  - 1
(%i2) find_root (f, x, 0, 2);
(%o2)                           .5533511515366971
(%i3) find_root (f, x, 0.8, 1.2);
(%o3)                           1.048019239841907
(%i4) find_root (f, x, 1.2, 2);
(%o4)                           1.399744786166543
```

从上面例子看出，`(%i2)`中`find_root`只返回了区间`[0, 2]`上的一个解，其中至少还有两个解，分别在区间`[0.8, 1.2]`和`[1.2, 2]`之上。

牛顿法 使用牛顿法来求方程数值解，需要先使用命令`load (newton1)`来加载软件包（Maxima可以通过软件包来扩展其功能，这时候要用`load`命令加载软件包）。然后，执行求解命令

`newton (f, x, x_0, eps)`，

其中`f`是要求解的函数，`x`是自变量，`x_0`是开始搜索的起点，`eps`是求解的精度。

```
(%i1) load (newton1);
(%o1)          /usr/share/maxima/5.19.2/share/numeric/newton1.mac
(%i2) newton (sin(x^2), x, 2, 1e-4);
(%o2)                          1.772455072439521
(%i3) sin (%^2);
(%o3)                       - 4.3302267947770734E-6
```

2.9 复数

在Maxima里复数由实部加上虚部乘以`%i`表示。之前介绍的那些算术运算及数学函数大都可以直接应用于复数。这里只介绍复数独特的运算。

2.9.1 实部与虚部

`realpart`和`imagpart`分别返回复数的实部和虚部。

```
(%i1) z: 5 + 3*%i $
(%i2) z^2;
                                          2
(%o2)                              (3 %i + 5)
(%i3) expand (%);
(%o3)                              30 %i + 16
(%i4) realpart (%);
(%o4)                                  16
(%i5) imagpart (%o3);
(%o5)                                  30
```

2.9.2 复共轭

```
(%i1) conjugate (a + %i*b);
(%o1)                              a - %i b
```

2.9.3 复数模和辐角

```
(%i1) z: 1 + %i;
(%o1)                                  %i + 1
(%i2) abs (z);
(%o2)                                 sqrt(2)
(%i3) carg (z);
                                        %pi
(%o3)                                   ---
                                         4
```

2.9.4 直角形式和极坐标形式

`rectform`和`polarform`分别返回复数的直角形式和极坐标形式。

```
(%i1) (10 + %i*5)/(3 + %i*4);
                                     5 %i + 10
(%o1)                               ----------
                                     4 %i + 3
(%i2) rectform (%);
(%o2)                                 2 - %i
(%i3) polarform (%);
                                       - %i  atan(1/2)
(%o3)                          sqrt(5) %e
```

3 矩阵

3.1 矩阵输入

3.1.1 交互式输入

使用`entermatrix (m, n)`函数可以进行交互式的矩阵输入，Maxima将每个元素一一读入。如果行列维数相同，Maxima会主动询问矩阵是否为对角、对称、反对称或者一般矩阵，这样可以有效减少输入次数。

```
(%i1) m: entermatrix (2, 2);

Is the matrix  1. Diagonal  2. Symmetric  3. Antisymmetric  4. General
Answer 1, 2, 3 or 4 :
2;
Row 1 Column 1:
a;
Row 1 Column 2:
a^2;
Row 2 Column 2:
1;

Matrix entered.
                                    [      2 ]
                                    [ a    a ]
(%o1)                               [        ]
                                    [   2    ]
                                    [ a    1 ]
```

3.1.2 以列表形式输入

使用`matrix (row_1, ..., row_n)`函数可以以列表形式将矩阵按行输入。

```
(%i1) x: matrix ([3, 6, 5], [2, 1, 2], [8, 5, 3]);
                                    [ 3  6  5 ]
                                    [         ]
(%o1)                               [ 2  1  2 ]
                                    [         ]
                                    [ 8  5  3 ]
```

3.1.3 以函数形式输入

使用`genmatrix (a, m, n)`函数，可以利用一个以元素角标为变量的函数`a [i, j]`自动生成一个$m \times n$矩阵。

```
(%i1) a [i, j] := (i - j + 1)^2;
                                                    2
(%o1)                          a      := (i - j + 1)
                                i, j
(%i2) genmatrix (a, 3, 3);
                                    [ 1   0   1 ]
                                    [           ]
(%o2)                               [ 4   1   0 ]
                                    [           ]
                                    [ 9   4   1 ]
```

注意：

- 这里的函数定义方法和前面第2.4节讲的一般函数不一样。这里是数组函数，自变量列表在方括号里面，而不是圆括号。

3.1.4 对角矩阵

对于具有相同元素的对角矩阵，还有更简便的输入方法。`diagmatrix (n, x)`函数返回一个对角元素为x的$n \times n$对角矩阵。单位矩阵可以用`diagmatrix (n, 1)`表示。另外，单位矩阵还可以通过`ident (n)`获得。

3.1.5 单一非零元素矩阵

如果一个$m \times n$矩阵只在第i行第j列含有一个非零元素x，可以用`ematrix (m, n, x, i, j)`函数获得。

3.2 矩阵运算

3.2.1 加减乘除

前面介绍的普通加减乘除运算（`+, -, *, /`）都可以应用于矩阵。这时候，运算分别作用于每个单独元素。

```
(%i1) m1: matrix ([2, 3], [4, 5]);
                                    [ 2   3 ]
(%o1)                               [       ]
                                    [ 4   5 ]
(%i2) m2: matrix ([6, 7], [3, 8]);
                                    [ 6   7 ]
(%o2)                               [       ]
                                    [ 3   8 ]
(%i3) m1 + m2;
                                    [ 8   10 ]
```

```
(%o3)                                   [          ]
                                        [ 7   13   ]
(%i4) m1 * m2;
                                       [ 12   21 ]
(%o4)                                  [         ]
                                       [ 12   40 ]
```

3.2.2 幂运算

当运算对象之一为标量时，幂运算也可以作用于矩阵。

```
(%i5) m1^3;
                                       [ 8    27   ]
(%o5)                                  [           ]
                                       [ 64   125  ]
(%i6) 3^m1;
                                       [ 9    27   ]
(%o6)                                  [           ]
                                       [ 81   243  ]
```

3.2.3 矩阵乘法

而对于矩阵乘法，需要使用点算符（也就是小数点.）。为了和小数点区分开，点算符（.）的前后应该留出空格。矩阵乘法是不满足交换律的。

```
(%i7) m1 . m2;
                                       [ 21   38 ]
(%o7)                                  [         ]
                                       [ 39   68 ]
```

3.3 行（列）操作

3.3.1 提取行（列）

用`row (M, i)`和`col (M, i)`函数分别提取矩阵`M`的第`i`行和第`i`列。

```
(%i1) M: matrix ([2, 3, 5], [7, 9, 3], [4, 8, 6]);
                                       [ 2   3   5 ]
                                       [           ]
(%o1)                                  [ 7   9   3 ]
                                       [           ]
                                       [ 4   8   6 ]
(%i2) row (M, 2);
(%o2)                                  [ 7   9   3 ]
```

```
(%i3) col (M, 1);
                                        [ 2 ]
                                        [   ]
(%o3)                                   [ 7 ]
                                        [   ]
                                        [ 4 ]
(%i4) (%o2) . (%o3);
(%o4)                                    89
```

3.3.2 增加行（列）

用`addcol (M, list_1, ..., list_n)`函数为矩阵`M`增加一列或多列。

```
(%i5) addcol (M, [11, 23, 16]);
                                  [ 2  3  5  11 ]
                                  [             ]
(%o5)                             [ 7  9  3  23 ]
                                  [             ]
                                  [ 4  8  6  16 ]
```

相应的行操作为`addrow (M, list_1, ..., list_n)`。

3.3.3 子矩阵

把一个矩阵的若干行和（或）列删除，就得到它的一个子矩阵。函数：

`submatrix (i_1, ..., i_m, M, j_1, ..., j_n)`

将矩阵`M`的第`i_1, ..., i_m`行和第`j_1, ..., j_m`列删除，返回一个新矩阵。

```
(%i1) a [i, j] := i + 2*j;
(%o1)                            a      := i + 2 j
                                  i, j
(%i2) M: genmatrix (a, 4, 4);
                                  [ 3  5  7   9  ]
                                  [              ]
                                  [ 4  6  8   10 ]
(%o2)                             [              ]
                                  [ 5  7  9   11 ]
                                  [              ]
                                  [ 6  8  10  12 ]
(%i3) submatrix (1, 2, M, 3);
                                    [ 5  7  11 ]
(%o3)                               [          ]
                                    [ 6  8  12 ]
```

3.4 线性代数

3.4.1 行列式

```
(%i1) M: matrix ([8, 1, 6], [3, 5, 7], [4, 9, 2]);
```

$$(\%o1)\quad \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$$

```
(%i2) determinant (M);
```

$$(\%o2)\quad -360$$

3.4.2 矩阵转置

```
(%i3) transpose (M);
```

$$(\%o3)\quad \begin{bmatrix} 8 & 3 & 4 \\ 1 & 5 & 9 \\ 6 & 7 & 2 \end{bmatrix}$$

3.4.3 矩阵的逆

```
(%i4) invert (M);
```

$$(\%o4)\quad \begin{bmatrix} \frac{53}{360} & -\frac{13}{90} & \frac{23}{360} \\ -\frac{11}{180} & \frac{1}{45} & \frac{19}{180} \\ -\frac{7}{360} & \frac{17}{90} & -\frac{37}{360} \end{bmatrix}$$

3.4.4 矩阵的秩

```
(%i5) rank (M);
```

$$(\%o5)\quad 3$$

3.4.5 高斯消元

`triangularize`函数使用高斯消元法返回一个梯形（或三角）矩阵。

```
(%i6) triangularize (M);
                                  [ 8   1      6    ]
                                  [                 ]
(%o6)                             [ 0   37    38    ]
                                  [                 ]
                                  [ 0   0    - 360  ]
```

`echelon`函数和`triangularize`功能类似，只不过`echelon`会把每行首项归一化。

```
(%i7) echelon (M);
                                  [      1   3   ]
                                  [ 1    -   -   ]
                                  [      8   4   ]
                                  [              ]
(%o7)                             [          38  ]
                                  [ 0    1   --  ]
                                  [          37  ]
                                  [              ]
                                  [ 0    0   1   ]
```

3.4.6 本征值和本征向量

`eigenvalues`函数返回一个列表，其中又包含两个列表元素：第一个是本征值，第二个是各本征值对应的重数。

```
(%i8) eigenvalues (M);
(%o8)                [[- 2 sqrt(6), 2 sqrt(6), 15], [1, 1, 1]]
```

`eigenvectors`函数也返回一个包含两个元素的列表：第一个其实就是`eigenvalues`的输出结果，第二个是对应于各本征值的本征向量。

```
(%i9) eigenvectors (M);
(%o9) [[[- 2 sqrt(6), 2 sqrt(6), 15], [1, 1, 1]],
                                   3/2
      2 sqrt(6) + 2     2     sqrt(3) + 7
[[[1, -------------, - ----------------]],
            5                  5
                                   3/2
       2 sqrt(6) - 2   2     sqrt(3) - 7
[[1, - -------------, ----------------]], [[1, 1, 1]]]]
            5                  5
```

4 微积分

4.1 微分

`diff (f(x), x, n)`返回函数`f(x)`的`n`阶导函数。
`diff (f(x, y, ...), x, n, y, m, ...)`返回函数`f(x, y, ...)`对于自变量`x, y, ...`的混合偏导函数。

```
(%i1) diff (sin(x)*exp(2*x), x, 2);
                                 2 x                  2 x
(%o1)                      3 %e     sin(x) + 4 %e     cos(x)
(%i2) diff ((x - 2*y)^5, x, 1, y, 2);
                                                      2
(%o2)                              240 (x - 2 y)
```

4.2 积分

4.2.1 不定积分

使用`integrate (f(x), x)`来进行不定积分。

```
(%i1) integrate (x/(1+x)^2, x);
                                                  1
(%o1)                              log(x + 1) + -----
                                                 x + 1
```

4.2.2 定积分

`integrate`函数也可以用来计算定积分。`integrate (f(x), x, a, b)`表示从`a`到`b`的积分限上函数`f(x)`的定积分。

```
(%i1) integrate (x*sin(x)*exp(-x), x, 0, %pi);
                                                 - %pi
                                 (%pi + 1) %e          1
(%o1)                            ------------------ + -
                                         2             2
(%i2) integrate(exp(-a*x^2), x, 0, inf);
Is  a  positive, negative, or zero?

positive;
                                       sqrt(%pi)
(%o2)                                  ---------
                                       2 sqrt(a)
```

由这里的例子可以看出，当由于参数不同而会得到不同结果时，Maxima会询问用户相应条件。

4.2.3 数值积分

Maxima使用一套叫做QUADPACK的函数集来做数值积分。对于一般函数`f(x)`，可以使用函数：

`quad_qag (f(x), x, a, b, key, [epsrel, epsabs, limit])`

来计算从`a`到`b`的数值积分。其中`key`是一个取值从1到6的整数，用于选择不同的积分器。较大的`key`值适合于做较强振荡函数的积分。`epsrel`、`epsabs`和`limit`都是用来控制计算误差的可选参数，其中`epsrel`是相对误差，`epsabs`是绝对误差，`limit`是计算时使用的最大子区间数。

`quad_qag`函数返回一个包含四个元素的列表，返回值分别是：

1. 积分值，
2. 绝对误差，
3. 被积函数计算次数，
4. 错误代码。

如果错误代码为0，说明计算过程中没有问题发生。如果为任何非0的数字，均说明碰到了问题。

```
(%i1) quad_qag (x/sqrt(16+x^2), x, 0, 3, 5, epsrel=1d-9);
(%o1)          [.9999999999999998, 1.110223024625157E-14, 51, 0]
(%i2) integrate (x/sqrt(16+x^2), x, 0, 3);
(%o2)                                  1
```

除了`quad_qag`之外，Maxima还有一些其他的QUADPACK数值积分函数：

- `quad_qags (f(x), x, a, b, [epsrel, epsabs, limit])`
  一般函数积分；
- `quad_qagi (f(x), x, a, b, [epsrel, epsabs, limit])`
  积分限为无穷或者半无穷的积分；
- `quad_qawc (f(x), x, c, a, b, [epsrel, epsabs, limit])`
  计算$f(x)/(x-c)$的柯西主值；
- `quad_qawf (f(x), x, a, omega, trig, [epsabs, limit, maxp1, limlst])`
  计算正弦或余弦傅立叶变换，即$\int_a^\infty f(x)\sin(\omega x)\,\mathrm{d}x$或者$\int_a^\infty f(x)\cos(\omega x)\,\mathrm{d}x$，其中`trig`的值为`sin`或`cos`；
- `quad_qawo (f(x), x, a, b, omega, trig, [epsrel, epsabs, limit, maxp1, limlst])`
  同上，只不过积分限是从`a`到`b`的有限区间；
- `quad_qaws (f(x), x, a, b, alpha, beta, wfun, [epsrel, epsabs, limit])`
  在有限区间上计算具有$f(x)w(x)$形式的特定代数或者对数函数的积分。`wfun`取值从1到4，对应四个加权函数：

1. $w(x) = (x-a)^{\alpha}(b-x)^{\beta}$;
2. $w(x) = (x-a)^{\alpha}(b-x)^{\beta}\log(x-a)$;
3. $w(x) = (x-a)^{\alpha}(b-x)^{\beta}\log(b-x)$;
4. $w(x) = (x-a)^{\alpha}(b-x)^{\beta}\log(x-a)\log(b-x)$。

若要更详细的了解上述函数，请参照Maxima官方手册。

4.3 泰勒展开

函数`f(x)`的在`x = a`附近的幂级数可以通过`powerseries (f(x), x, a)`获得。

```
(%i1) powerseries (1/(1-x^2), x, 0);
                                  inf
                                  ====
                                  \        2 i1
(%o1)                              >      x
                                  /
                                  ====
                                  i1 = 0
```

但是很多时候我们无法得到级数的解析表示，这时候可以用`taylor (f(x), x, a, n)`得到函数`f(x)`在`x = a`附近第`n`阶项（`(x - a)^n`）以下各项的泰勒级数。

```
(%i1) taylor (sqrt(1 - sin(x)), x, 0, 5);
                                 2     3     4       5
                            x    x     x     x       x
(%o1)/T/                1 - - - -- + -- + --- - ---- + . . .
                            2    8    48   384    3840
```

4.4 拉普拉斯变换

`laplace (f(t), t, s)`计算函数`f(t)`的拉普拉斯变换，其中`t`为自变量，`s`为变换参量。相应的反变换为`ilt (f(s), s, t)`。

```
(%i1) laplace (exp (2*t + a) * sin(t) * t, t, s);
                                   a
                                %%e   (2 s - 4)
(%o1)                          ----------------
                                 2            2
                               (s  - 4 s + 5)
(%i2) ilt (%, s, t);
                                 2 t + a
(%o2)                        t %%e         sin(t)
```

4.5 留数

`residue (f(z), z, z_0)`计算`f(z)`在`z = z_0`点的留数。

```
(%i1) residue (sin(a*x)/x^4, x, 0);
```

(%o1) $$-\frac{a^3}{6}$$

5 微分方程

5.1 一阶或二阶常微分方程通解

ode2 (eqn, dvar, ivar)函数用来解一阶或者二阶常微分方程，其中eqn是待解方程，dvar是因变量，ivar是自变量。

```
(%i1) eq: 'diff(y, x, 2) - y = 4*x*sin(x);
```

$$(\%o1)\quad \frac{d^2 y}{dx^2} - y = 4\,x\,\sin(x)$$

```
(%i2) ode2(eq, y, x);
```

$$(\%o2)\quad y = -2\,x\,\sin(x) - 2\,\cos(x) + \%k1\,\%e^{x} + \%k2\,\%e^{-x}$$

这里得到了常微分方程的一个通解，其中%k1和%k2是任意常数。

注意：

- 在定义方程时，微分函数*diff*之前有一个单引号（'），这表示让*Maxima*只给出形式上的输出，并不真的进行计算。这是因为我们这里只要列出方程，并不想让*Maxima*真的求导。
- 之前我们介绍的那些函数命令基本上都可以用这种方式阻止它们执行运算，只给出形式上的结果。这在*Maxima*里叫做名词化，也就是用单引号阻止动词（函数）执行。

5.2 初值问题

函数ic1 (solution, xval, yval)和ic2 (solution, xval, yval, dval)分别用来解一阶和二阶微分方程的初值问题，其中solution是用ode2解得的通解，xval和yval分别是自变量和因变量的初值，dval是因变量一阶导数的初值。

```
(%i3) ic2 (%, x=0, y=1, 'diff(y, x)=2);
```

$$(\%o3)\quad y = -2\,x\,\sin(x) - 2\,\cos(x) + \frac{5\,\%e^{x}}{2} + \frac{\%e^{-x}}{2}$$

5.3 边值问题

函数bc2 (solution, xval_1, yval_1, xval_2, yval_2)用来求解二阶微分方程的边值问题，其中solution是ode2解得的通解，xval_1、yval_1、xval_2和yval_2分别为自变量和因变量在第一点和第二点的取值。

```
(%i4) bc2 (%o2, x=0, y=1, x=%pi, y=2);
```

$$(\%o4)\qquad y = -2\,x\,\sin(x) - 2\,\cos(x) - \frac{3\,\%e^{x}}{\%e^{2\,\%pi} - 1} + \frac{3\,\%e^{2\,\%pi - x}}{\%e^{2\,\%pi} - 1}$$

5.4 一阶线性微分方程组

函数`desolve ([eqn_1, ..., eqn_n], [x_1, ..., x_n])`使用拉普拉斯变换求解一阶线性微分方程组，其中`eqn_1...`是待解方程，`x_1...`是待解函数。这里待解函数不能只写变量名（例如`y`），而需要明确写出对自变量的依赖关系（例如`y(x)`）。如果初值是已知的，可以使用`atvalue`命令来提供初值。

```
(%i1) eqn1: 'diff(x(t), t) - 3*x(t) + 2*y(t) = cos(t);
```

$$(\%o1)\qquad \frac{d}{dt}\,(x(t)) + 2\,y(t) - 3\,x(t) = \cos(t)$$

```
(%i2) eqn2: 'diff(y(t), t) - 2*x(t) + y(t) = 0;
```

$$(\%o2)\qquad \frac{d}{dt}\,(y(t)) + y(t) - 2\,x(t) = 0$$

```
(%i3) atvalue (x(t), t=0, 0);
```

$$(\%o3)\qquad 0$$

```
(%i4) atvalue (y(t), t=0, 1);
```

$$(\%o4)\qquad 1$$

```
(%i5) desolve ([eqn1, eqn2], [x(t), y(t)]);
```

$$(\%o5)\qquad [x(t) = -\frac{\sin(t)}{2} - \frac{\cos(t)}{2} - t\,\%e^{t} + \frac{\%e^{t}}{2},\ y(t) = -\sin(t) - t\,\%e^{t} + \%e^{t}]$$

```
(%i6) ratsimp (%o5);
```

$$(\%o6)\qquad [x(t) = -\frac{\sin(t) + \cos(t) + (2\,t - 1)\,\%e^{t}}{2},\ y(t) = (1 - t)\,\%e^{t} - \sin(t)]$$

6 特殊函数

Maxima提供有常用的特殊函数。这里不介绍每个函数的具体用法，可以参考任何一本数理方程教材，这里只列表如下：

表达式	函数名
`bessel_j(`ν, z`)`	ν阶第一类贝塞尔函数
`bessel_y(`ν, z`)`	ν阶第二类贝塞尔函数
`bessel_i(`ν, z`)`	ν阶变形（虚宗量）第一类贝塞尔函数
`bessel_k(`ν, z`)`	ν阶变形（虚宗量）第二类贝塞尔函数
`hankel_1(`ν, z`)`	ν阶第一类汉克尔函数
`hankel_2(`ν, z`)`	ν阶第二类汉克尔函数
`gamma(`z`)`	Γ函数（伽马函数）
`beta(`a, b`)`	B函数（贝塔函数）
`erf(`z`)`	误差函数

如果使用`load (orthopoly)`加载`orthopoly`外部包，还可以提供以下常用函数：

表达式	函数名
`hermite(`n, x`)`	厄米多项式
`laguerre(`n, x`)`	拉盖尔多项式
`legendre_p(`n, x`)`	第一类勒让德多项式
`legendre_q(`n, x`)`	第二类勒让德多项式
`assoc_legendre_p(`n, m, x`)`	n次m阶第一类连带勒让德函数
`assoc_legendre_q(`n, m, x`)`	n次m阶第二类连带勒让德函数
`spherical_bessel_j(`n, x`)`	第一类球贝塞尔函数
`spherical_bessel_y(`n, x`)`	第二类球贝塞尔函数
`spherical_hankel1(`n, x`)`	第一类球汉克尔函数
`spherical_hankel2(`n, x`)`	第二类球汉克尔函数
`spherical_harmonic(`n, m, x, y`)`	球谐函数

7 作图

Maxima调用外部程序来实现作图，默认的外部程序是Gnuplot。Gnuplot是一个很强大的基于命令行的函数及数据作图程序，集成了计算、拟合、脚本编程等功能，包括Maxima，Octave等在内的一些软件均使用Gnuplot作为后台程序实现作图功能。对于那些经常用到数学作图的用户，我建议直接使用Gnuplot，因为它有更灵活的设置和更强大的功能。对于偶尔需要在Maxima环境下作图的用户，Maxima提供了一些基本功能。这里做一个简单介绍。

7.1 二维作图

7.1.1 一般函数作图

二维作图的基本命令是`plot2d`，它的基本使用方式是：

`plot2d (y(x), [x, min_x, max_x], [y, min_y, max_y], options, ...)`

也就是函数表达式，加上变量取值范围，再加上一些控制作图的参数列表。y的取值范围不是必需的。常用的控制参数我们在下面逐步介绍。先看一个例子：

例1：

```
(%i1) plot2d (sin(x), [x, -5, 5], [y, -1.2, 1.2]) $
```

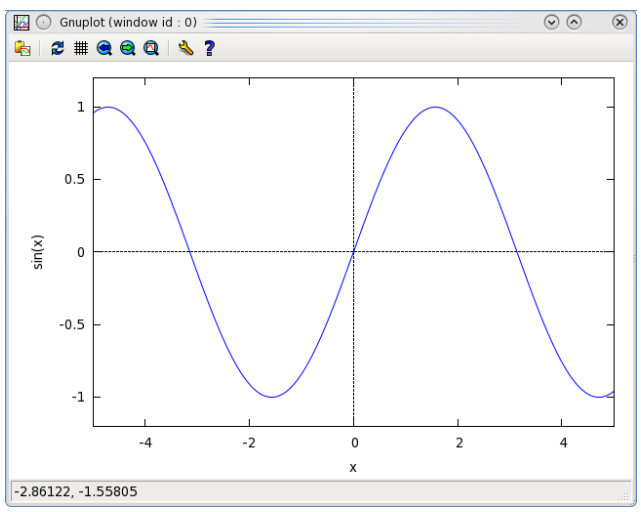

图 1: 二维做图：一般函数作图（例1）

若同时有几个函数，`plot2`的使用方式是：

`plot2d ([y_1(x), y_2(x), ...], [x, min_x, max_x], ..., options, ...)`

这时候几个函数被放在了方括号中，也就是一个列表里。

例2：

```
(%i1) plot2d ([sin(x), cos(x)], [x, -5, 5], [y, -1.2, 1.2]) $
```

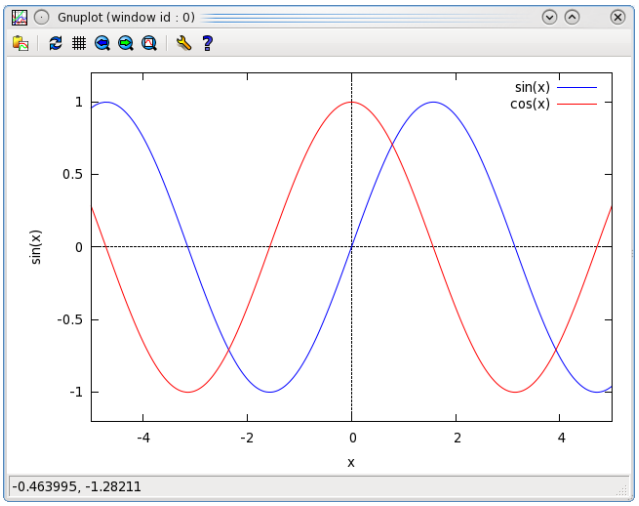

图 2: 二维做图：多个函数（例2）

7.1.2 对数坐标图

例3：

```
(%i1) plot2d (exp(x^2), [x, 0, 3], [logy]) $
```

如果同时加上`[logx]`，就是双对数坐标图了。

7.1.3 参数方程作图

例4：

```
(%i5) plot2d ([parametric, cos(t), sin(3*t), [t, 0, 2*%pi]], [x, -1.2, 1.2],
  [y, -1.2, 1.2], [nticks, 200]) $
```

如果在函数列表之前加上`parametric`，就代表参数方程作图。`nticks`参数控制画图时函数计算取样的数目。如果这个值过小，可能画出的函数曲线不够平滑。

7.1.4 数据作图

预先把数据存于列表之中，然后可以使用`discrete`参数作图。

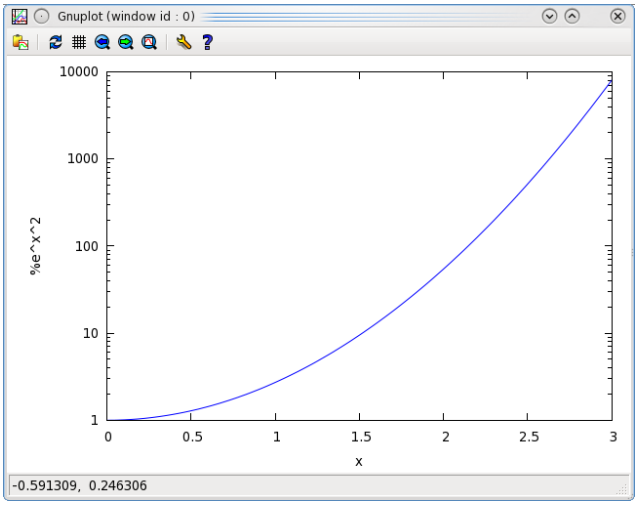

图 3: 二维做图：对数坐标（例3）

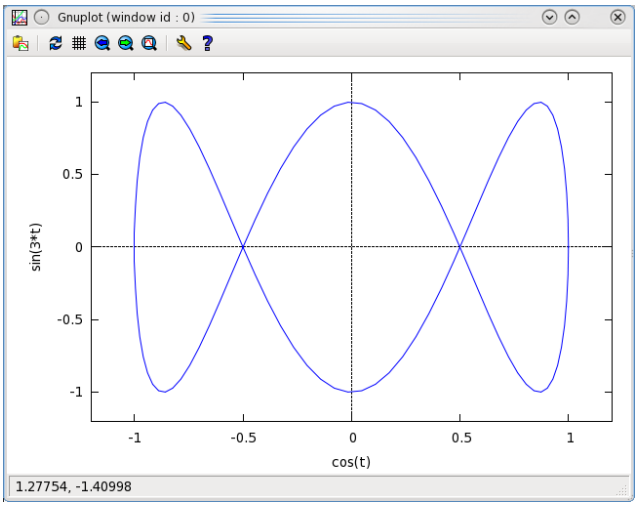

图 4: 二维做图：参数方程作图（例4）

例5：

```
(%i1) x: [0, 1, 2, 3, 4, 5, 6] $
(%i2) y: [8.9, 4.1, 0.9, 0.1, 1.1, 3.8, 9.1] $
(%i3) plot2d ([discrete, x, y], [style, linespoints], [xlabel, "Time"],
  [ylabel, "Displacement"]) $
```

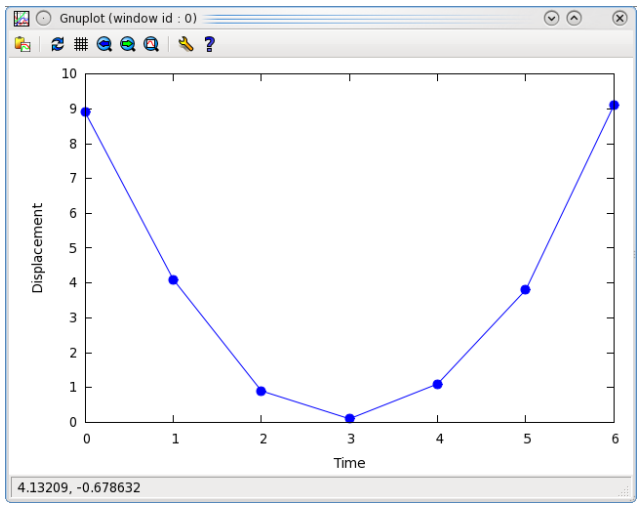

图 5: 二维作图：数据作图（例5）

参数`style`可以接受`linespoints`，`lines`，`points`等画图方式。`xlabel`和`ylabel`分别设置横纵坐标的标签。

7.1.5 存为图片文件

如果想把作图结果存为一个图片文件，例如png文件，可以使用下面参数：
`[gnuplot_term, png], [gnuplot_out_file, "graph.png"]`

7.2 三维作图

三维函数作图和二维非常类似，只不过多了一个自变量，命令变为`plot3d`。

例6：

```
(%i1) plot3d ([cos(x)*(3 + y*cos(x/2)), sin(x)*(3 + y*cos(x/2)), y*sin(x/2)],
  [x, -%pi, %pi], [y, -1, 1], ['grid, 50, 15]) $
```

这个例子里有三个函数，两个自变量，这其实是莫比乌斯带的参数方程。

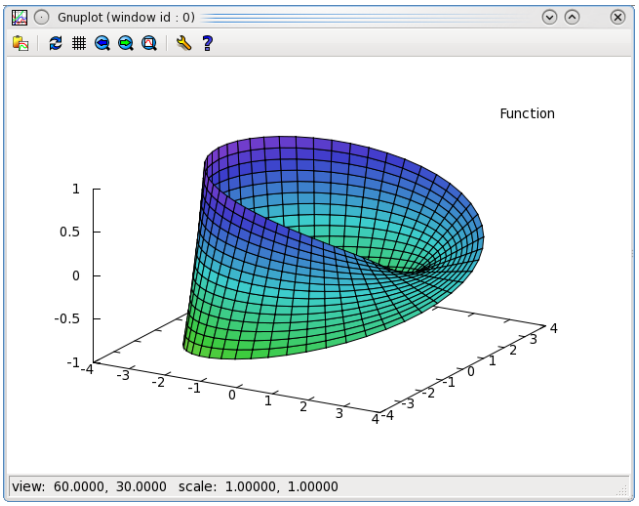
Gnuplot (window id : 0)
Function
1
0.5
0
-0.5
-1
-4
-3
-2
-1
0
1
2
3
4
-4
-3
-2
-1
0
1
2
3
4
view: 60.0000, 30.0000 scale: 1.00000, 1.00000

图 6: 三维做图：（例6）

8 图形界面

Maxima可以在终端界面下完成几乎所有任务，但是有时候我们还是希望有一个图形界面。这不仅仅是为了更赏心悦目，更重要的原因是：当一个输出结果非常复杂时，仅靠文字终端界面很难看的清楚。这时候如果有图形化输出的话，可以对复杂的结果一目了然。另外，菜单选项对于那些还不太熟悉Maxima命令的使用者多少有点提示作用。

这里只列举出四个最常用的Maxima图形界面，他们各有特点，使用者可以根据自己的情况进行选择。它们的具体使用方法这里不作介绍，大家可以根据自己的操作系统查找相应的资料。

8.1 wxMaxima

xwMaxima是一个基于wxWidgets的跨平台图形界面。它支持二维数学公式显示，可以通过菜单选择大部分Maxima命令。

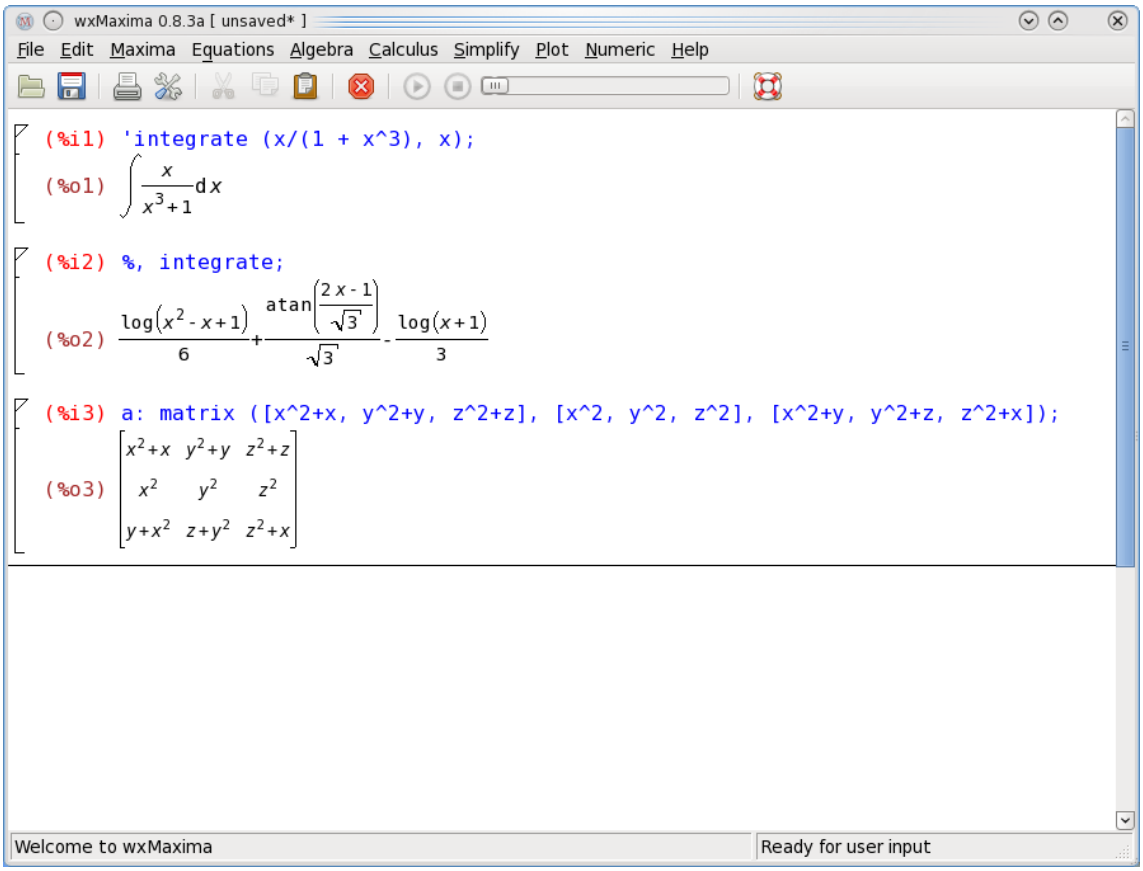

图 7: wxMaxima显示效果

8.2 xmaxima

官方的Maxima图形界面，基于Tcl/Tk。xmaxima没有提供二维公式显示和菜单输入命令功能，但是它有一个浏览器，可以随时浏览帮助文件。

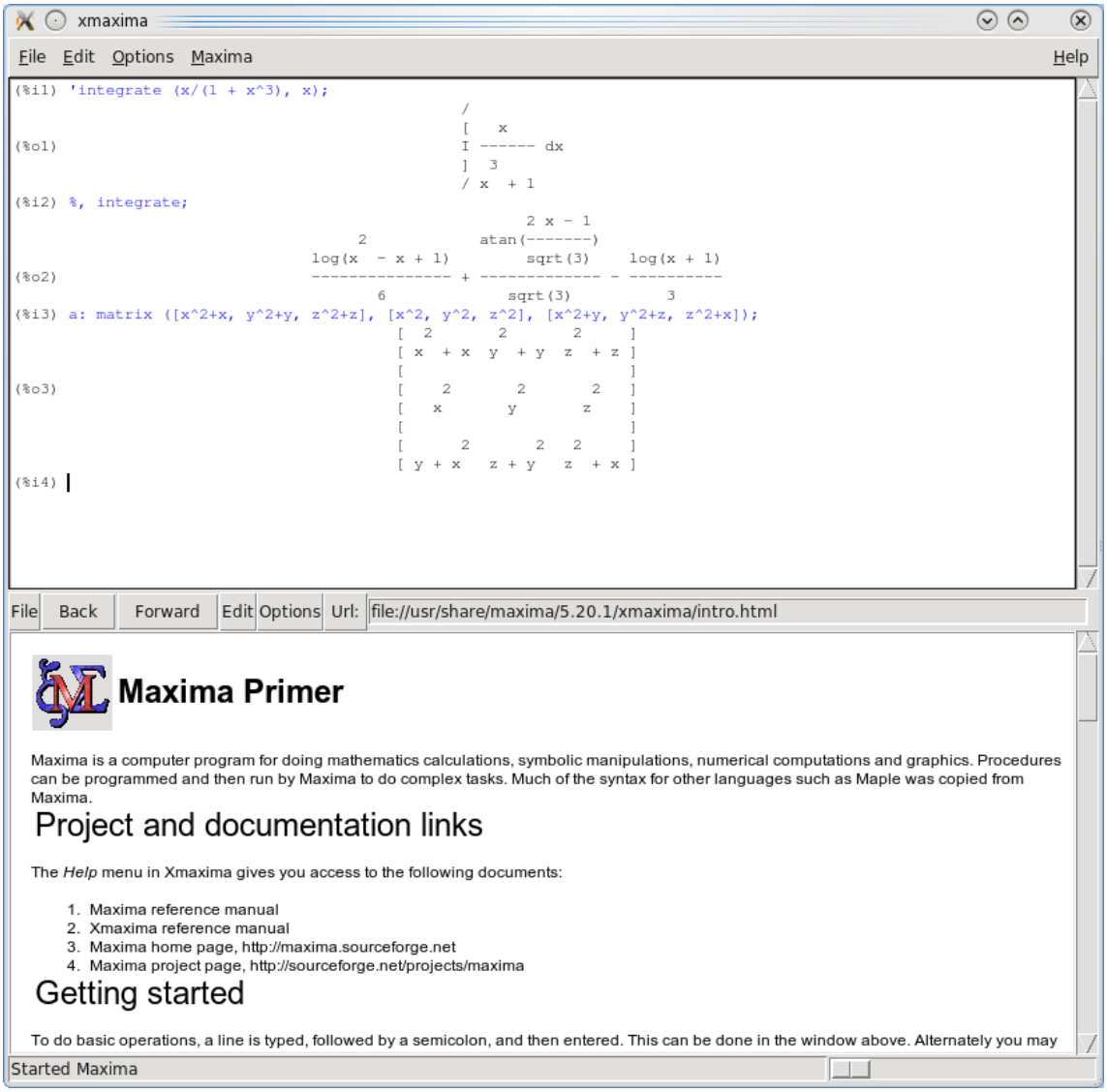

图 8: xmaxima显示效果

8.3 TeXmacs

TeXmacs是一个WYSIWYW（所见即所得）的科技文档编辑器，但是它同时可以用作很多程序的前端，例如Maxima，Gnuplot等等，所以它很适合在同一个文档中混排各种程序的运行结果。由于TeXmacs使用TeX字体，所以它可能是公式显示效果最好的Maxima图形界面。

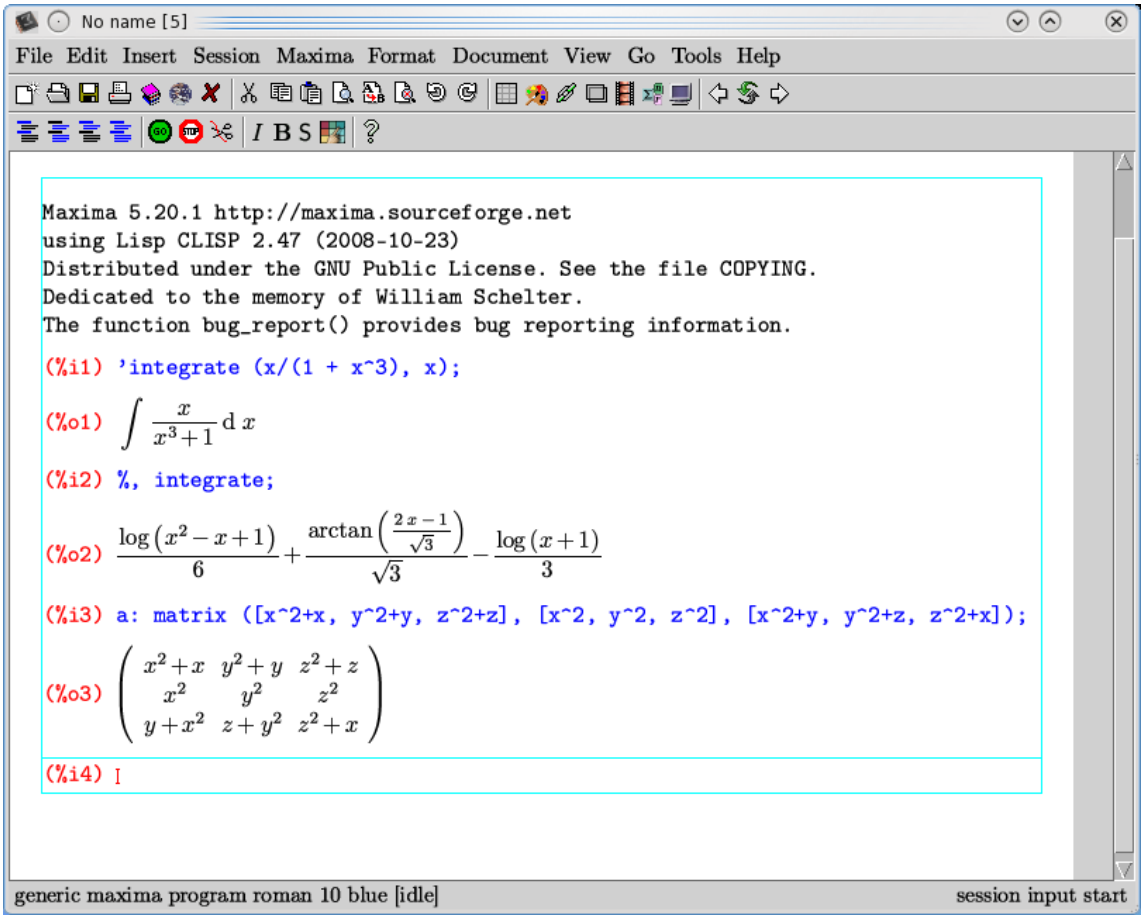

图 9: TeXmacs显示效果

8.4 Emacs+imaxima

Emacs配合imaxima模式也能达到非常好的显示效果。如果你已经熟悉Emacs，不妨采用这种方式。

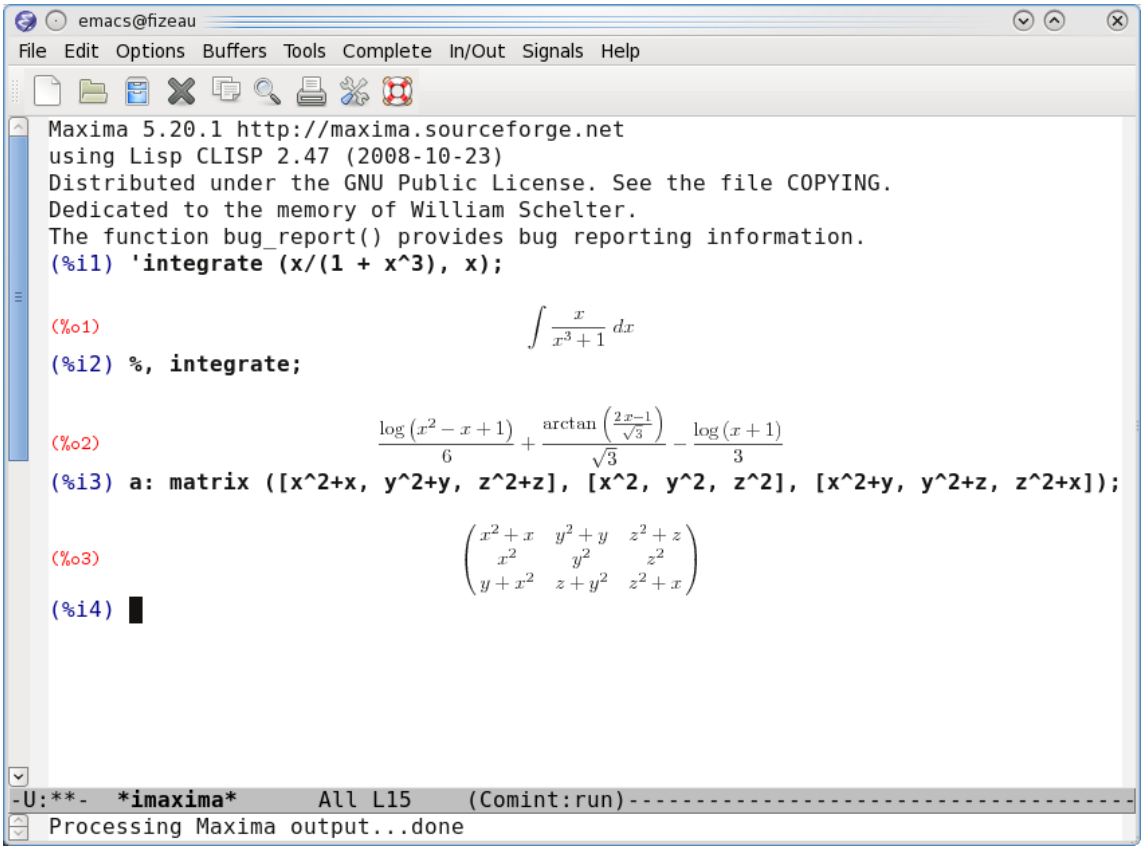

图 10: Emacs+imaxima模式显示效果

9 Maxima编程

Maxima的流程控制命令允许把命令预先编辑好，自动调用执行。

9.1 do循环

do循环语句有下面几种常用形式:

- `for 变量: 初值 step 步长 thru 界限 do 循环体`
- `for 变量: 初值 step 步长 while 条件 do 循环体`
- `for 变量: 初值 step 步长 unless 条件 do 循环体`
- `for 变量: in 列表 do 循环体`

当步长为1时，可以省去step这一项。

```
(%i1) for a:0 step 3 thru 10 do display(a) $
                                        a = 0

                                        a = 3

                                        a = 6

                                        a = 9

(%i2) s: 0 $
(%i3) for i: 1 while i <= 10 do s: s+i;
(%o3)                                   done
(%i4) s;
(%o4)                                    55
(%i5) for f in [sin, cos, tan] do print(f, "(%pi/4) = ", f(%pi/4)) $
                    1
sin (%pi/4) =  -------
               sqrt(2)
                    1
cos (%pi/4) =  -------
               sqrt(2)
tan (%pi/4) =  1
```

注意：

- 上面例子中的*display*和*print*命令都用于显示计算结果。*display*对于每一个结果都分行显示，而*print*可以在同一行显示多个结果。
- *while* $i <= 10$和*unless* $i > 10$其实是等效的。

9.2 if条件句

`if`条件语句形式:

- `if` 条件 `then` 表达式1 `else` 表达式0
- `if` 条件1 `then` 表达式1 `elseif` 条件2 `then` 表达式2 `elseif ... else` 表达式0

Maxima里`if`语句的构成形式和其他编程语言很类似，所以不多做说明。“条件”对应逻辑运算和关系判别式，Maxima里的关系和逻辑算符列表如下:

名称	符号
小于	`<`
小于等于	`<=`
大于	`>`
大于等于	`>=`
等于（全等）	`=`
不等于（`=`的否定式）	`#`
等于（数值相等）	`equal(a, b)`
不等于（`equal`的否定式）	`notequal(a, b)`
与	`and`
或	`or`
非	`not`

注意：

- 上面表中的“=”和“*equal(a, b)*”虽然都是相等，但是略有不同：
  - 用法上：“=”用于两个表达式之间，而“*equal(a, b)*”是一个函数，被比较的表达式是函数变量；
  - 意义上：“=”表示两个表达式完全相同，而“*equal(a, b)*”只是表示数值相等。看下面的例子：

```
(%i1) f: x^2 - 1 $
(%i2) g: (x + 1)*(x - 1) $
(%i3) is (f = g);
(%o3)                                false
(%i4) is (equal(f, g));
(%o4)                                true
```

上面`is`命令用于判断一个判别句的真假。

9.3 block程序块

顾名思义，block就是把若干命令放在一起，构成一个相对独立的环境。block可以定义一些局部变量，而一般Maxima里面的变量都是全局的。block的使用方法是：

block ([变量1, 变量2, ...], 表达式1, 表达式2, ...)

其中在列表中定义的那些变量只在block里有效。block的返回值等于最后一个表达式的值。

9.4 read读取

read命令会从用户那里读取一个输入，然后继续运行程序。

```
(%i1) block ( [a, b], a: read ("a = "), b: read ("b = "),
print ("a + b = ", a + b)) $
a =
5;
b =
7;
a + b =  12
```

9.5 结果输出

9.5.1 二维表示

Maxima默认的输出为二维表示，这样结果中的表达式可以一目了然。但是有时候我们还是希望得到一维结果，例如需要把输出结果再用作输入时。display2d变量就是控制输出结果的二维表示的，它的默认值为true。

```
(%i1) diff (x/sqrt(x^2-1), x);
                                                  2
                                                 x
(%o1)                    ------------ - ------------
                            2              2      3/2
                         sqrt(x  - 1)    (x  - 1)
(%i2) display2d:false$

(%i3) diff (x/sqrt(x^2-1), x);

(%o3) 1/sqrt(x^2-1)-x^2/(x^2-1)^(3/2)
```

9.5.2 输出到文件

对于上面输出的一维结果，如果我们想在将来什么时候再使用它，最好把它存到一个文件里备用。stringout可以把一个或多个表达式按照一维方式存入文件，它的基本使用方式为：

`stringout ("文件名", 表达式1, 表达式2, ...)`

9.6 随机数

使用`random (x)`可以产生一个伪随机数。如果`x`是整数，则返回值为0到`x-1`之间的整数；如果`x`是浮点数，则返回值为小于`x`的非负浮点数。

9.7 batch执行程序

如果预先把要执行的Maxima命令存在一个文件里，则可以从Maxima里用`batch (文件名)`来调用该程序。

9.8 tex输出

使用`tex`命令可以把Maxima表达式输出为TEX代码，这样用户可以很方便的把计算结果插入到自己的LATEX文档中。例如：

```
(%i1) integrate (1/(1+x^3), x);
                                                 2 x - 1
                                              atan(-------)
                        2                        sqrt(3)
                   log(x  - x + 1)                                  log(x + 1)
(%o1)            - ---------------- + ------------------- + ----------
                         6                   sqrt(3)              3
(%i2) tex (integrate (1/(1+x^3), x)) $
$$-{{\log \left(x^2-x+1\right)}\over{6}}+{{\arctan \left({{2\,x-1
 }\over{\sqrt{3}}}\right)}\over{\sqrt{3}}}+{{\log \left(x+1\right)
 }\over{3}}$$
```

LATEX根据上面例子中Maxima提供的TEX代码生成的公式：

$$-\frac{\log\left(x^2-x+1\right)}{6}+\frac{\arctan\left(\frac{2\,x-1}{\sqrt{3}}\right)}{\sqrt{3}}+\frac{\log\left(x+1\right)}{3}$$

索 引

*, 8
**, 8
+, 8
-, 8
/, 8
:, 6
:=, 8
;, 4
<, 40
<=, 40
=, 40
>, 40
>=, 40
?, 5
??, 5
#, 40
$, 6
%, 6
%e, 8
%i, 8
%pi, 8
^, 8

. , 19

abs, 8, 15
acos, 8
acosh, 8
acot, 8
acoth, 8
acsc, 8
acsch, 8
addcol, 20
addrow, 20
allroots, 14
and, 40
asec, 8
asech, 8
asin, 8
asinh, 8
assoc_legendre_p, 29
assoc_legendre_q, 29
assume, 11
atan, 8
atanh, 8
atvalue, 28

batch, 42
bc2, 27
bessel_i, 29
bessel_j, 29
bessel_k, 29
bessel_y, 29
beta, 29
bfloat, 6
block, 41

carg, 15
col, 19
conjugate, 15
cos, 8
cosh, 8
cot, 8
coth, 8
csc, 8
csch, 8

desolve, 28
determinant, 21
diagmatrix, 18
diff, 23
display, 39
display2d, 41
do, 39

echelon, 22
eigenvalues, 22
eigenvectors, 22
else, 40
elseif, 40
ematrix, 18
entermatrix, 17
equal, 40
erf, 29
example, 5
exp, 8
expand, 9, 10

factor, 10
facts, 12
find_root, 14
float, 6
for, 39
forget, 12
fpprec, 6

gamma, 29
genmatrix, 17

hankel_1, 29
hankel_2, 29
hermite, 29

ic1, 27
ic2, 27
ident, 18
if, 40
ilt, 25
imagpart, 15
in, 39
inf, 8
infinity, 8
integrate, 23
invert, 21
is, 40

laguerre, 29
laplace, 25
legendre_p, 29
legendre_q, 29
let, 12
letsimp, 12
load, 15
log, 8

matrix, 17
max, 8
min, 8
minf, 8

newton, 15
not, 40
notequal, 40
numer, 6

ode2, 27
or, 40
orthopoly, 29

plot2d, 30
plot3d, 33
polarform, 16
powerseries, 25
print, 39
product, 9

quad_qag, 24
quad_qagi, 24
quad_qags, 24
quad_qawc, 24
quad_qawf, 24
quad_qawo, 24
quad_qaws, 24
quit, 4

radcan, 11
random, 42

rank, 21
ratexpand, 10
ratsimp, 10
read, 41
realpart, 15
realroots, 14
rectform, 16
residue, 26
row, 19

sec, 8
sech, 8
sign, 8
simpproduct, 9
simpsum, 9
sin, 8
sinh, 8
solve, 13
spherical_bessel_j, 29
spherical_bessel_y, 29
spherical_hankel1, 29
spherical_hankel2, 29
spherical_harmonic, 29
sqrt, 8
step, 39
stringout, 41
submatrix, 20
subst, 10
sum, 8

tan, 8
tanh, 8
taylor, 25
tex, 42
then, 40
thru, 39
transpose, 21
triangularize, 22
trigexpand, 12
triginverses, 13
trigrat, 12
trigreduce, 12
trigsimp, 12

unless, 39

while, 39