Introduction to Maxima
맥시마 길라잡이

Richard H. Rand
Dept. of Theoretical and Applied Mechanics, Cornell University [*]
양사장 애벌 번역

05-04-2010

[*]Adapted from "Perturbation Methods, Bifurcation Theory and Computer Algebra" by Rand and Armbruster, Springer, 1987. Adapted to LaTeX and HTML by Nelson L. Dias (nldias@simepar.br), SIMEPAR Technological Institute and Federal University of Paraná, Brazil. Updated by Robert Dodier, August 2005. Updated by Jaime Villate and Sajang Yang, April 2010. Translated into Korean by Sajang Yang, May 2010.

Contents

1 Introduction 길라잡이

To invoke Maxima in a console, type
화면에서 맥시마를 실행시키려면, 다음과 같이 입력하셔요.

```
maxima <enter>
```

The computer will display a greeting of the sort:
다음과 같이 출력됩니다:

```
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
This is a development version of Maxima. The function bug_report()
provides bug reporting information.
(%i1)
```

The `(%i1)` is a "label". Each input or output line is labelled and can be referred to by its own label for the rest of the session. `i` labels denote your commands and `o` labels denote displays of the machine's response. *Never use variable names like* `%i1` *or* `%o5`*, as these will be confused with the lines so labeled.*

`(%i1)`은 레이블입니다. 각 입출력에 레이블을 달아 세션이 살아있는 동안 참조할 수 있는 것이지요. `i`로 시작하는 레이블은 명령(입력)을 나타내고, `o`로 시작하는 레이블은 명령에 대한 응답(출력)을 나타내는 데 씁니다.
`%i1`나 `%o5`처럼 생긴 변수 이름은 절대 쓰지 마셔요. 맥시마 헛갈려요.

Maxima distinguishes lower and upper case. All built-in functions have names which are lowercase only (`sin`, `cos`, `save`, `load`, etc). Built-in constants have lowercase names (`%e`, `%pi`, `inf`, etc). If you type `SIN(x)` or `Sin(x)`, Maxima assumes you mean something other than the built-in `sin` function. User-defined functions and variables can have names which are lower or upper case or both. `foo(XY)`, `Foo(Xy)`, `FOO(xy)` are all different.

맥시마는 영문 대·소문자를 구별합니다. `sin`, `cos`, `save`, `load` 등등 내장 함수는 오로지 소문자로만 되어있구요. `%e`, `%pi`, `inf`처럼 미리 정의된 상수들도 소문자로 되어있습니다. `SIN(x)`나 `Sin(x)`를 명령하면 맥시마는 내장 함수인 `sin`이 아닌 딴 함수로 여길 겁니다. 사용자 정의 함수와 변수는 대문자나 소문자 아님 짬뽕으로도 쓸 수 있습니다. `foo(XY)`, `Foo(Xy)`, `FOO(xy)`은 다 비슷해 보이지만 다 다른 함수와 변수입니다.

2 Special keys and symbols 특수 키와 기호

1. To end a Maxima session, type `quit();`.
   맥시마 세션을 종료하고 싶을 때는, `quit();`를 쳐주셔요.

2. To abort a computation without leaving Maxima, type `^C`. (Here `^` stands for the control key, so that `^C` means first press the key marked control and hold it down

while pressing the C key.) It is important for you to know how to do this in case, for example, you begin a computation which is taking too long. For example:
맥시마 세션을 종료하지 않고 그냥 현재 진행중인 작업만 중지시키고 싶을 때는 `^C`를 눌러주셔요. (여기서 우산 표시 `^`은 PC 키보드 중에서 "Ctrl"이라고 적혀있는 컨트롤 키를 꾸욱 누르고 있는 것을 말합니다. "Ctrl"키는 왼쪽 아래에 하나, 또 오른쪽 아래에 하나 있습니다. '뭘 눌러야하나' 긴장하지 마시고 둘 중 아무거나 눌러도 됩니다. 이 컨트롤 키를 누른 상태에서 C 키를 누르는 것을 `^C`로 표시합니다. 보통 사람들이 왼쪽 키를 많이 누르니 특이하게 보이시려면 오른쪽 키를 눌러주시고요, 그래도 잘 모르시겠으면 옆에 PC 쓰시는 분한테 물어 보셔요.) 무지 오래 걸리는 계산을 걸어놓다가 중간에 중지하고 싶을 때 쓰시면 좋습니다. 예를 들자면,

```
(%i1) sum (1/x^2, x, 1, 10000);
^C
Maxima encountered a Lisp error:

 Interactive interrupt at #xA70E274.

Automatically continuing.
To enable the Lisp debugger set *debugger-hook* to nil.
(%i2)
```

3. In order to tell Maxima that you have finished your command, use the semicolon (`;`), followed by a return. Note that the return key alone does not signal that you are done with your input.
명령(입력)의 끝에는 항상 세미콜론을 붙입니다. 그리고 키보드에 'RETURN' 또는 'Enter'라고 써있는 키를 반드시 눌러야 됩니다. 이 세미콜론을 빼먹든 지, 엔터키 안 누르고 모니터만 째려본다고 해서 답 안나옵니다.

4. An alternative input terminator to the semicolon (`;`) is the dollar sign (`$`), which, however, supresses the display of Maxima's computation. This is useful if you are computing some long intermediate result, and you don't want to waste time having it displayed on the screen.
세미콜론(;)이 싫으면 딸라 표시($)를 쓰셔도 됩니다. 계산이 되긴 되는 데, 계산 결과를 안 보여줍니다. 긴 프로그램 짤 때는 중간중간 결과 보는 게 성가시기도 하고 시간 낭비도 되지요. 바로 '고럴 때 고로코롬' 쓰기에 좋습니다.

5. If you wish to repeat a command which you have already given, say on line `(%i5)`, you may do so without typing it over again by preceding its label with two single quotes (`''`), i.e., `''%i5`. (Note that simply inputing `%i5` will not do the job — try it.)
이미 입력한 명령을 또 사용하고 싶을 때, 말하자면 `(%i5)`의 명령을 다시 입력하지 않고 실행시키고 싶으면 작은 따옴표 두 개(`''`)를 레이블 앞에 붙여주시면 되지요, 즉, `''%i5`처럼 말입니다. 작은 따옴표 두 개 빼먹고 레이블 `%i5`만 쳐넣으면 안됩니다. 못 믿으시면, 한 번 해보셔요. (속고만 살았나?)

6. If you want to refer to the immediately preceding result computed by Maxima, you can either use its `o` label, or you can use the special symbol percent (`%`).

맥시마가 바로 앞에 계산한 결과를 참조하고 싶을 때에는 특수 기호인 퍼센트 기호 (%)를 쓰시든 지, 아니면 그 결과에 해당하는 o 레이블을 쓰셔도 됩니다.

7. The standard quantities e (natural log base), i (square root of -1) and π (3.14159...) are respectively referred to as `%e`, `%i`, and `%pi`. Note that the use of `%` here as a prefix is completely unrelated to the use of `%` to refer to the preceding result computed.
   자연 로그 밑수인 e, (-1의 제곱근인) i, 그리고 π (3.14159...)와 같은 표준 값들은 `%e`, `%i`, 그리고 `%pi`로 나타냅니다. 여기서 쓴 `%`는 앞에서 결과값 참조를 위해서 사용한 `%`랑은 전혀 상관없습니다. 주의하시고요.

8. In order to assign a value to a variable, Maxima uses the colon (`:`), not the equal sign. The equal sign is used for representing equations.
   변수에 값을 지정할 때, 맥시마는 등호(`=`) 말고 콜론(`:`)을 씁니다, 등호는 방정식을 나타낼 때 씁니다.

3 Arithmetic 산수

The common arithmetic operations are
자주 쓰는 산술 연산자는

`+` 덧셈 (addition)

`-` 뺄셈 (subtraction)

`*` 스칼라곱 (scalar multiplication)

`/` 나눗셈 (division)

^

ı∗∗ø(*exponentiation*), (*matrixmultiplication*)

`x`의 제곱근 (square root of).
Maxima's output is characterized by exact (rational) arithmetic. For example,
맥시마는 완전한 (분수) 모양으로 출력합니다. 예를 들자면,

```
(%i1) 1/100 + 1/101;
                                   201
(%o1)                             -----
                                  10100
```

If irrational numbers are involved in a computation, they are kept in symbolic form:
무리수가 있으면, 고것들은 그냥 기호 형태로 냅둡니다:

```
(%i2) (1 + sqrt(2))^5;
                                             5
(%o2)                           (sqrt(2) + 1)
(%i3) expand(%);
                                   7/2
(%o3)                           3 2    + 5 sqrt(2) + 41
```

However, it is often useful to express a result in decimal notation. This may be accomplished by following the expression you want expanded by "`,numer`":
무시로 계산 결과를 숫자로 표시하는 게 더 쓸모 있을 때가 생깁니다. 그럴 때는 표현식 뒤에다가 "`,numer`"를 추가합니다:

```
(%i4) %, numer;
(%o4)                          82.01219330881977
```

Note the use here of `%` to refer to the previous result. In this version of Maxima, `numer` gives 16 significant figures, of which the last is often unreliable. However, Maxima can offer *arbitrarily high precision* by using the `bfloat` function:
위에서 쓰인 `%`는 바로 그 앞에 계산된 결과를 나타냅니다. 주의하셔요. 위의 예에서 맥시마는 16자리 유효숫자를 썼는 데요, 그 걸로는 종종 정밀도가 부족하지요. 맥시마는 `bfloat` 함수를 써서 아무 정밀도나 제공합니다:

```
(%i5) bfloat (%o3);
(%o5)                  8.201219330881976B1
```

The number of significant figures displayed is controlled by the Maxima variable `fpprec`, which has the default value of 16:
출력되는 유효숫자 갯수는 맥시마 내장 변수 `fpprec`에 의해 결정됩니다.

```
(%i6) fpprec;
(%o6)                          16
```

Here we reset `fpprec` to yield 100 digits:
밑에 `fpprec`를 100으로 바꿔봅니다:

```
(%i7) fpprec: 100;
(%o7)                          100
(%i8) ''%i5;
(%o8) 8.20121933088197564152489730020812442785204843859314941221\
237124017312418754011041266612384955016056lb1
```

Note the use of two single quotes (`''`) in `(%i8)` to repeat command `(%i5)`. Maxima can handle very large numbers without approximation:
`(%i8)`의 작은 따옴표 두개 (`''`)는 명령 `(%i5)`를 반복하라는 것입니다. 맥시마는 근사화 없이 아주 큰 숫자를 다룰 수 있습니다:

```
(%i9) 100!;
(%o9) 9332621544394415268169923885626670049071596826438162146859\
2963895217599993229915608941463976156518286253697920827223758251\
185210916864000000000000000000000000
```

4 Algebra 대수

Maxima's importance as a computer tool to facilitate analytical calculations becomes more evident when we see how easily it does algebra for us. Here's an example in which

a polynomial is expanded:
해석적 계산을 쉽게 하는 컴퓨터 도구로서 맥시마의 중요성은 앨지브라를 풀 때 명확해집니다. 다항식이 전개될 때 예를 들어보겠습니다:

```
(%i1) (x + 3*y + x^2*y)^3;
                              2             3
(%o1)                     (x  y + 3 y + x)
(%i2) expand (%);
       6  3      4  3       2  3       3      5  2       3  2
(%o2) x  y  + 9 x  y  + 27 x  y  + 27 y  + 3 x  y  + 18 x  y
                                          2      4        2      3
                                + 27 x y  + 3 x  y + 9 x  y + x
```

Now suppose we wanted to substitute `5/z` for `x` in the above expression:
이제 위 수식에 대하여 `x`를 `5/z`로 치환하려고 할 때:

```
(%i3) %o2, x=5/z;
            2        3                  2                3
      135 y    675 y    225 y   2250 y    125   5625 y    1875 y
(%o3) ------ + ------ + ----- + ------- + --- + ------- + ------
        z         2       2        3       3       4         4
                 z       z        z       z       z         z
                                                 2          3
                                           9375 y    15625 y        3
                                         + ------- + -------- + 27 y
                                              5          6
                                             z          z
```

The Maxima function `ratsimp` will place this over a common denominator:
맥시마 함수 `ratsimp`는 위 전개식을 공통분모로 보여줍니다:

```
(%i4) ratsimp (%);
            3  6        2  5          3            4
(%o4) (27 y  z  + 135 y  z  + (675 y  + 225 y) z
           2          3          3            2         2
 + (2250 y  + 125) z  + (5625 y  + 1875 y) z  + 9375 y  z
           3   6
 + 15625 y )/z
```

Expressions may also be `factor`ed:
또 수식은 인수분해 (`factor`) 가능하고요.

```
(%i5) factor (%);
                                    2              3
                         (3 y z  + 5 z + 25 y)
(%o5)                    ----------------------
                                     6
                                    z
```

Maxima can obtain exact solutions to systems of nonlinear algebraic equations. In this example we `solve` three equations in the three unknowns `a`, `b`, `c`:
맥시마는 비선형 방정식의 완전해를 구할 수도 있습니다. 다음 예에서 세 변수 `a`, `b`, `c` 를 구하는 세 방정식을 풀어봅니다:

```
(%i6) a + b*c = 1;
(%o6)                      b c + a = 1
(%i7) b - a*c = 0;
(%o7)                      b - a c = 0
(%i8) a + b = 5;
(%o8)                       b + a = 5
(%i9) solve ([%o6, %o7, %o8], [a, b, c]);
            25 sqrt(79) %i + 25      5 sqrt(79) %i + 5
(%o9) [[a = -------------------, b = -----------------,
            6 sqrt(79) %i - 34       sqrt(79) %i + 11
    sqrt(79) %i + 1        25 sqrt(79) %i - 25
c = ---------------], [a = -------------------,
          10               6 sqrt(79) %i + 34
    5 sqrt(79) %i - 5        sqrt(79) %i - 1
b = -----------------, c = - ---------------]]
    sqrt(79) %i - 11               10
```

Note that the display consists of a "list", i.e., some expression contained between two brackets `[ ...]`, which itself contains two lists. Each of the latter contain a distinct solution to the simultaneous equations.
"리스트"가 출력됩니다, 즉 `[ ...]` 꺽쇠 괄호 안에 여러 표현식이 있는 데, 여기서는 다시 리스트 두 개를 가집니다. 각각 위 연립 방정식의 해가 됩니다.

Trigonometric identities are easy to manipulate in Maxima. The function `trigexpand` uses the sum-of-angles formulas to make the argument inside each trig function as simple as possible:
삼각 함수는 누워서 떡먹기입니다. 내장 함수 `trigexpand` 덧셈 정리를 이용해 삼각 함수 표현을 가능한 간단히 합니다.

```
(%i10) sin(u + v) * cos(u)^3;
                                 3
(%o10)                        cos (u) sin(v + u)
(%i11) trigexpand (%);
                  3
(%o11)         cos (u) (cos(u) sin(v) + sin(u) cos(v))
```

The function `trigreduce`, on the other hand, converts an expression into a form which is a sum of terms, each of which contains only a single `sin` or `cos`:
반대로 `trigreduce`함수는 삼각 함수 표현식을 하나의 `sin`이나 `cos`함수 가지는 각 항의 합으로 표현합니다:

```
(%i12) trigreduce (%o10);
```

```
       sin(v + 4 u) + sin(v - 2 u)   3 sin(v + 2 u) + 3 sin(v)
(%o12) --------------------------- + -------------------------
                    8                            8
```

The functions `realpart` and `imagpart` will return the real and imaginary parts of a complex expression:
`realpart`와 `imagpart`는 복소수 표현식의 실수와 허수 성분을 각각 리턴합니다:

```
(%i13) w: 3 + k*%i;
(%o13)                      %i k + 3
(%i14) w^2 * %e^w;
                                2   %i k + 3
(%o14)                (%i k + 3)  %e
(%i15) realpart (%);
                 3       2                3
(%o15)         %e  (9 - k ) cos(k) - 6 %e  k sin(k)
```

5 Calculus
미적분

Maxima can compute derivatives and integrals, expand in Taylor series, take limits, and obtain exact solutions to ordinary differential equations. We begin by defining the symbol `f` to be the following function of `x`:
맥시마는 미분과 적분, 테일러 시리즈, 극한, 상미분 방정식의 해를 구할 수 있습니다. `x`의 함수를 `f`라 하는 것에서부터 시작해 봅니다:

```
(%i1) f: x^3 * %e^(k*x) * sin(w*x);
                         3   k x
(%o1)                   x  %e    sin(w x)
```

We compute the derivative of `f` with respect to `x`:
`f`를 `x`에 대해서 미분합니다:

```
(%i2) diff (f, x);
         3   k x                2   k x
(%o2) k x  %e    sin(w x) + 3 x  %e    sin(w x)
                                                  3   k x
                                           + w x  %e    cos(w x)
```

Now we find the indefinite integral of `f` with respect to `x`:
`f`를 `x`에 대하여 적분합니다:

```
(%i3) integrate (f, x);
            6      3  4      5  2    7   3
(%o3) (((k w  + 3 k  w  + 3 k  w  + k ) x
       6      2  4      4  2      6   2
 + (3 w  + 3 k  w  - 3 k  w  - 3 k ) x
```

```
            4       3  2      5         4       2  2      4
 + (- 18 k w  - 12 k  w  + 6 k ) x - 6 w  + 36 k  w  - 6 k )
   k x                 7      2  5      4  3    6     3
 %e    sin(w x) + ((- w  - 3 k  w  - 3 k  w  - k  w) x
         5       3  3      5     2
 + (6 k w  + 12 k  w  + 6 k  w) x
       5       2  3      4              3       3      k x
 + (6 w  - 12 k  w  - 18 k  w) x - 24 k w  + 24 k  w) %e
             8      2  6      4  4      6  2    8
 cos(w x))/(w  + 4 k  w  + 6 k  w  + 4 k  w  + k )
```

A slight change in syntax gives definite integrals:
약간 바꿔서 정적분을 계산해 봅니다:

```
(%i4) integrate (1/x^2, x, 1, inf);
(%o4)                          1
(%i5) integrate (1/x, x, 0, inf);

defint: integral is divergent.
 -- an error. To debug this try: debugmode(true);
```

Next we define the symbol `g` in terms of `f` (previously defined in `%i1`) and the hyperbolic sine function, and find its Taylor series expansion (up to, say, order 3 terms) about the point `x = 0`:
다음은 심볼(함수) g를 `f` (`f`는 `%i1`에서 정의되었습니다.), 하이퍼볼릭 사인으로 정의해봅니다. 그리고 `x = 0`일 때, 테일러 시리즈 3차 계수까지 전개해 봅니다.

```
(%i6) g: f / sinh(k*x)^4;
                           3   k x
                          x  %e    sin(w x)
(%o6)                     -----------------
                                 4
                             sinh (k x)
(%i7) taylor (g, x, 0, 3);
                               2    3   2          2    3   3
             w    w x   (w k  + w ) x    (3 w k  + w ) x
(%o7)/T/     -- + --- - -------------- - ---------------- + . . .
              4    3            4                  3
             k    k          6 k                6 k
```

The limit of `g` as `x` goes to 0 is computed as follows:
x가 0으로 접근할 때, g의 극한 값은 다음과 같습니다:

```
(%i8) limit (g, x, 0);
                                w
(%o8)                           --
                                 4
                                k
```

Maxima also permits derivatives to be represented in unevaluated form (note the quote):
또한 평가되지 않은 미분 형태로도 나타낼 수 있습니다:

```
(%i9) 'diff (y, x);
                                dy
(%o9)                           --
                                dx
```

The quote operator in `(%i9)` means "do not evaluate". Without it, Maxima would have obtained 0:
`(%i9)`의 따옴표는 "평가하지 마세요"라는 뜻입니다. 그게 없으면, 0이라고 답이 나왔겠지요:

```
(%i10) diff (y, x);
(%o10)                           0
```

Using the quote operator we can write differential equations:
따옴표 연산자를 이용해서 미분 방정식을 쓸 수 있습니다:

```
(%i11) 'diff (y, x, 2) + 'diff (y, x) + y;
                            2
                           d y   dy
(%o11)                     --- + -- + y
                             2   dx
                           dx
```

Maxima's `ode2` function can solve first and second order ODE's:
내장 함수 `ode2`를 이용하여 일차 및 이차 상미분 방정식을 풀 수 있습니다:

```
(%i12) ode2 (%o11, y, x);
                   - x/2          sqrt(3) x              sqrt(3) x
(%o12)       y = %e      (%k1 sin(---------) + %k2 cos(---------))
                                      2                      2
```

6 Matrix calculations 행렬 연산

Maxima can compute the determinant, inverse and eigenvalues and eigenvectors of matrices which have symbolic elements (i.e., elements which involve algebraic variables.) We begin by entering a matrix `m` element by element:
맥시마는 행렬의 디터미넌트, 역행렬, 고유값, 고유벡터를 구할 수 있습니다. 행렬 `m`의 각 항을 일일이 입력하며 시작해 봅니다:

```
(%i1) m: entermatrix (3, 3);

Is the matrix  1. Diagonal  2. Symmetric  3. Antisymmetric  4. General
Answer 1, 2, 3 or 4 :
4;
Row 1 Column 1:
0;
Row 1 Column 2:
1;
Row 1 Column 3:
a;
Row 2 Column 1:
1;
Row 2 Column 2:
0;
Row 2 Column 3:
1;
Row 3 Column 1:
1;
Row 3 Column 2:
1;
Row 3 Column 3:
0;

Matrix entered.
                                  [ 0  1  a ]
                                  [         ]
(%o1)                             [ 1  0  1 ]
                                  [         ]
                                  [ 1  1  0 ]
```

Next we find its transpose, determinant and inverse:
전치 행렬, 디터미넌트, 역행렬을 구합니다:

```
(%i2) transpose (m);
                                  [ 0  1  1 ]
                                  [         ]
(%o2)                             [ 1  0  1 ]
```

```
                                  [         ]
                                  [ a  1  0 ]
(%i3) determinant (m);
(%o3)                               a + 1
(%i4) invert (m), detout;
                               [ - 1   a    1  ]
                               [               ]
                               [  1   - a   a  ]
                               [               ]
                               [  1    1   - 1 ]
(%o4)                          -----------------
                                    a + 1
```

In (%i4), the modifier detout keeps the determinant outside the inverse. As a check, we multiply m by its inverse (note the use of the period to represent matrix multiplication):
(%i4) 명령에서, detout 모디파이어는 디터미넌트를 역행렬의 밖에 놔둡니다. 시험삼아, 행렬 m을 그 역행렬로 곱해봅니다. (마침표는 행렬곱을 나타냅니다.):

```
(%i5) m . %o4;
                                     [ - 1   a    1  ]
                                     [               ]
                                     [  1   - a   a  ]
                      [ 0  1  a ]    [               ]
                      [         ]    [  1    1   - 1 ]
(%o5)                 [ 1  0  1 ] . -----------------
                      [         ]          a + 1
                      [ 1  1  0 ]
(%i6) expand (%);
       [   a       1                                   ]
       [ ----- + -----        0              0         ]
       [ a + 1   a + 1                                 ]
       [                                               ]
       [                   a       1                   ]
(%o6)  [       0         ----- + -----       0         ]
       [                 a + 1   a + 1                 ]
       [                                               ]
       [                                   a       1   ]
       [       0              0          ----- + ----- ]
       [                                 a + 1   a + 1 ]
(%i7) factor (%);
                                  [ 1  0  0 ]
                                  [         ]
(%o7)                             [ 0  1  0 ]
                                  [         ]
                                  [ 0  0  1 ]
```

In order to find the eigenvalues and eigenvectors of m, we use the function eigenvectors:
행렬 m의 고유치와 고유값을 구하려면, 내장함수 eigenvectors를 사용합니다:

```
(%i8) eigenvectors (m);
           sqrt(4 a + 5) - 1  sqrt(4 a + 5) + 1
(%o8) [[[- -----------------, -----------------, - 1],
                   2                  2
                   sqrt(4 a + 5) - 1    sqrt(4 a + 5) - 1
[1, 1, 1]], [[[1, - -----------------, - -----------------]],
                        2 a + 2              2 a + 2
     sqrt(4 a + 5) + 1  sqrt(4 a + 5) + 1
[[1, -----------------, -----------------]], [[1, - 1, 0]]]]
          2 a + 2            2 a + 2
```

In `%o8`, the first triple gives the eigenvalues of `m` and the next gives their respective multiplicities (here each is unrepeated). The next three triples give the corresponding eigenvectors of `m`. In order to extract from this expression one of these eigenvectors, we may use the `part` function:
`%o8`에서 첫번째 3중 꺽쇠 괄호의 리스트는 고유값을 그 다음은 그 곱입니다. 그 다음 3중 꺽쇠 괄호는 행렬 `m`의 고유벡터이고요, 그 중에서 따로 뽑으려면 내장함수 `part`를 씁니다:

```
(%i9) part (%o23, 2, 1, 1);
                  sqrt(4 a + 5) - 1    sqrt(4 a + 5) - 1
(%o9)     [1, - -----------------, - -----------------]
                       2 a + 2              2 a + 2
```

7 Programming in Maxima 맥시마 프로그래밍

So far, we have used Maxima in the interactive mode, rather like a calculator. However, for computations which involve a repetitive sequence of commands, it is better to execute a program. Here we present a short sample program to calculate the critical points of a function `f` of two variables `x` and `y`. The program cues the user to enter the function `f`, then it computes the partial derivatives f_x and f_y, and then it uses the Maxima command `solve` to obtain solutions to $f_x = f_y = 0$. The program is written outside of Maxima with a text editor, and then loaded into Maxima with the `batch` command. Here is the program listing:
여태까지는 계산기처럼 인터랙티브 모드만 사용하였지요. 하지만, 이제 반복적인 일련의 명령을 수행하려하면 프로그램을 실행시키는 것이 더 좋을 것입니다. 변수 `x`와 `y`를 갖는 함수 `f`의 임계점을 계산하는 프래그램을 보셔요. 프로그램은 사용자에게 함수 `f`를 입력하도록 하고나서, `solve` 함수를 이용해 $f_x = f_y = 0$의 해를 구합니다. 아무 텍스트 편집기로 작성하여 맥시마에 `batch` 명령으로 로드됩니다. 아래 리스팅을 보셔요:

```
/* --------------------------------------------------------------------------
   this is file critpts.max:
   as you can see, comments in maxima are like comments in C

   Nelson Luis Dias, nldias@simepar.br
   created 20000707
   updated 20000707
   -------------------------------------------------------------------------- */
critpts():=(
   print("program to find critical points"),
/* --------------------------------------------------------------------------
   asks for a function
   -------------------------------------------------------------------------- */
   f:read("enter f(x,y)"),
/* --------------------------------------------------------------------------
   echoes it, to make sure
   -------------------------------------------------------------------------- */
   print("f = ",f),
/* --------------------------------------------------------------------------
   produces a list with the two partial derivatives of f
   -------------------------------------------------------------------------- */
   eqs:[diff(f,x),diff(f,y)],
/* --------------------------------------------------------------------------
   produces a list of unknowns
   -------------------------------------------------------------------------- */
   unk:[x,y],
/* --------------------------------------------------------------------------
   solves the system
   -------------------------------------------------------------------------- */
   solve(eqs,unk)
)$
```

The program (which is actually a function with no argument) is called `critpts`. Each line is a valid Maxima command which could be executed from the keyboard, and which is separated by the next command by a comma. The partial derivatives are stored in a variable named `eqs`, and the unknowns are stored in `unk`. Here is a sample run:
프로그램 이름은 critpts입니다. 각 줄마다 맥시마 명령을 적고, '콤마'로 다음 명령과 구분짓습니다. 편미분된 결과들이 변수 eqs에 저장되고, 임계점들은 unk에 저장됩니다. 실행을 시켜봅니다:

```
(%i1) batch ("critpts.max");

batching #p/home/robert/tmp/maxima-clean/maxima/critpts.max
(%i2) critpts() := (print("program to find critical points"),
f : read("enter f(x,y)"), print("f = ", f),
eqs : [diff(f, x), diff(f, y)], unk : [x, y], solve(eqs, unk))
(%i3) critpts ();
program to find critical points
enter f(x,y)
%e^(x^3 + y^2)*(x + y);
                 2    3
                y  + x
f =  (y + x) %e
(%o3) [[x = .4588955685487001 %i + .3589790871086935,
y = .4942017368275118 %i - .1225787367783657],
[x = .3589790871086935 - .4588955685487001 %i,
y = - .4942017368275118 %i - .1225787367783657],
[x = .4187542327234816 %i - .6923124204420268,
y = 0.455912070111699 - .8697262692814121 %i],
[x = - .4187542327234816 %i - .6923124204420268,
y = .8697262692814121 %i + 0.455912070111699]]
```

8 A partial list of Maxima functions 맥시마 함수 (쬐끔만...)

See the Maxima reference manual `doc/html/maxima_toc.html` (under the main Maxima installation directory). From Maxima itself, you can use `describe(`*`function name`*`)`. 자세한 것은 맥시마 참조 매뉴얼을 보셔요. `doc/html/maxima_toc.html` 디렉토리에 저장되어 있습니다. 맥시마 프롬프트에서는 `describe(`*`function name`*`)`를 사용해 보셔요.[1]

`allroots(a)` Finds all the (generally complex) roots of the polynomial equation `A`, and lists them in `numer`ical format (i.e. to 16 significant figures).

`append(a,b)` Appends the list `b` to the list `a`, resulting in a single list.

`batch(a)` Loads and runs a program with filename `a`.

`coeff(a,b,c)` Gives the coefficient of `b` raised to the power `c` in expression `a`.

`concat(a,b)` Creates the symbol `ab`.

`cons(a,b)` Adds `a` to the list `b` as its first element.

`demoivre(a)` Transforms all complex exponentials in `a` to their trigonometric equivalents.

`denom(a)` Gives the denominator of `a`.

`depends(a,b)` Declares `a` to be a function of `b`. This is useful for writing unevaluated derivatives, as in specifying differential equations.

`desolve(a,b)` Attempts to solve a linear system `a` of ODE's for unknowns `b` using Laplace transforms.

`determinant(a)` Returns the determinant of the square matrix `a`.

`diff(a,b1,c1,b2,c2,...,bn,cn)` Gives the mixed partial derivative of `a` with respect to each `bi`, `ci` times. For brevity, `diff(a,b,1)` may be represented by `diff(a,b)`. `'diff(...)` represents the unevaluated derivative, useful in specifying a differential equation.

`eigenvalues(a)` Returns two lists, the first being the eigenvalues of the square matrix `a`, and the second being their respective multiplicities.

[1] 역자 주: 아래 내용은 한글 번역에 포함되지 않았습니다. 참조 매뉴얼을 활용해주셔요.

`eigenvectors(a)` Does everything that `eigenvalues` does, and adds a list of the eigenvectors of `a`.

`entermatrix(a,b)` Cues the user to enter an `a` × `b` matrix, element by element.

`ev(a,b1,b2,...,bn)` Evaluates `a` subject to the conditions `bi`. In particular the `bi` may be equations, lists of equations (such as that returned by `solve`), or assignments, in which cases `ev` "plugs" the `bi` into `a`. The `Bi` may also be words such as `numer` (in which case the result is returned in numerical format), `detout` (in which case any matrix inverses in `a` are performed with the determinant factored out), or `diff` (in which case all differentiations in `a` are evaluated, i.e., `'diff` in `a` is replaced by `diff`). For brevity in a manual command (i.e., not inside a user-defined function), the `ev` may be dropped, shortening the syntax to `a,b1,b2,...,bn`.

`expand(a)` Algebraically expands `a`. In particular multiplication is distributed over addition.

`exponentialize(a)` Transforms all trigonometric functions in `a` to their complex exponential equivalents.

`factor(a)` Factors `a`.

`freeof(a,b)` Is true if the variable `a` is not part of the expression `b`.

`grind(a)` Displays a variable or function `a` in a compact format. When used with `writefile` and an editor outside of Maxima, it offers a scheme for producing `batch` files which include Maxima-generated expressions.

`ident(a)` Returns an `a` × `a` identity matrix.

`imagpart(a)` Returns the imaginary part of `a`.

`integrate(a,b)` Attempts to find the indefinite integral of `a` with respect to `b`.

`integrate(a,b,c,d)` Attempts to find the indefinite integral of `a` with respect to `b`. taken from `b` = `c` to `b` = `d`. The limits of integration `c` and `d` may be taken is `inf` (positive infinity) of `minf` (negative infinity).

`invert(a)` Computes the inverse of the square matrix `a`.

`kill(a)` Removes the variable `a` with all its assignments and properties from the current Maxima environment.

`limit(a,b,c)` Gives the limit of expression `a` as variable `b` approaches the value `c`. The latter may be taken as `inf` of `minf` as in `integrate`.

`lhs(a)` Gives the left-hand side of the equation `a`.

`loadfile(a)` Loads a disk file with filename `a` from the current default directory. The disk file must be in the proper format (i.e. created by a `save` command).

`makelist(a,b,c,d)` Creates a list of `a`'s (each of which presumably depends on `b`), concatenated from `b` = `c` to `b` = `d`

`map(a,b)` Maps the function `a` onto the subexpressions of `b`.

`matrix(a1,a2,...,an)` Creates a matrix consisting of the rows `ai`, where each row `ai` is a list of `m` elements, `[b1, b2, ..., bm]`.

`num(a)` Gives the numerator of `a`.

`ode2(a,b,c)` Attempts to solve the first- or second-order ordinary differential equation `a` for `b` as a function of `c`.

`part(a,b1,b2,...,bn)` First takes the `b1`th part of `a`, then the `b2`th part of that, and so on.

`playback(a)` Displays the last `a` (an integer) labels and their associated expressions. If `a` is omitted, all lines are played back. See the Manual for other options.

`ratsimp(a)` Simplifies `a` and returns a quotient of two polynomials.

`realpart(a)` Returns the real part of `a`.

`rhs(a)` Gives the right-hand side of the equation `a`.

`save(a,b1,b2,..., bn)` Creates a disk file with filename `a` in the current default directory, of variables, functions, or arrays `bi`. The format of the file permits it to be reloaded into Maxima using the `loadfile` command. Everything (including labels) may be `saved` by taking `b1` equal to `all`.

`solve(a,b)` Attempts to solve the algebraic equation `a` for the unknown `b`. A list of solution equations is returned. For brevity, if `a` is an equation of the form `c` = `0`, it may be abbreviated simply by the expression `c`.

`string(a)` Converts `a` to Maxima's linear notation (similar to Fortran's) just as if it had been typed in and puts `a` into the buffer for possible editing. The `string`'ed expression should not be used in a computation.

`stringout(a,b1,b2,...,bn)` Creates a disk file with filename `a` in the current default directory, of variables (e.g. labels) `bi`. The file is in a text format and is not reloadable into Maxima. However the strungout expressions can be incorporated into a Fortran, Basic or C program with a minimum of editing.

`subst(a,b,c)` Substitutes `a` for `b` in `c`.

`taylor(a,b,c,d)` Expands `a` in a Taylor series in `b` about `b` = `c`, up to and including the term $(\mathtt{b} - \mathtt{c})^{\mathtt{d}}$. Maxima also supports Taylor expansions in more than one independent variable; see the Manual for details.

`transpose(a)` Gives the transpose of the matrix `a`.

`trigexpand(a)` Is a trig simplification function which uses the sum-of-angles formulas to simplify the arguments of individual `sin`'s or `cos`'s. For example, `trigexpand(sin(x+y))` gives `cos(x) sin(y) + sin(x) cos(y)`.

`trigreduce(a)` Is a trig simplification function which uses trig identities to convert products and powers of `sin` and `cos` into a sum of terms, each of which contains only a single `sin` or `cos`. For example, `trigreduce(sin(x)^2)` gives `(1 - cos(2x))/2`.

`trigsimp(a)` Is a trig simplification function which replaces `tan`, `sec`, etc., by their `sin` and `cos` equivalents. It also uses the identity $\texttt{sin}()^2 + \texttt{cos}()^2 = \texttt{1}$.

The below is not the integral part of the original document.

p.s. by Sajang Yang: If you want to help improving this translation, please let me know through email: Sajang.Yang (at) gmail (dot) com. This TEX document is compiled with TexLive 2009, kotex package.

역자 후기: 번역에 불만이 있으신 분들은 Sajang.Yang (at) gmail (dot) com으로 이멜 날려 주세요. 이 TEX문서는 kotex 패키지와 함께 TeXLive 2009로 작성하였습니다.