An Introduction to Using MAXIMA

Why use MAXIMA as against performing numerical computations?
A quotation by R.W. Hamming:

"The purpose of computing is insight, not numbers"

Some of the capabilities of MAXIMA:

Limits	Taylor Series (several variables)
Derivatives	Poisson Series
Indefinite Integration	Laplace Transforms
Definite Integration	Indefinite Summation
Ordinary Differential Equations	Matrix Manipulation
Systems of Equations (non-linear)	Vector Manipulation
Simplification	Tensor Manipulation
Factorization	Fortran Generation

Sanjiva Weerawarana, September 05, 1995.

Starting and Stopping MAXIMA:

To use MAXIMA, you need to either type

```
/usr/local/maxima/maxima
```

or set up an alias for this purpose. To set up an alias in a csh/tcsh type environment, put the following in your .cshrc:

```
alias maxima /usr/local/maxima/maxima
```

If you're on **poseidon**, then you will need to type in the following to get MAXIMA to start:

```
(run)
```

To quit from MAXIMA, type `^D` (Control-D) or `quit();`

Differentiation:

Simple, one variable case:

```
(c1)   diff(f(x),x);

                                          d
(d1)                                      -- (f(x))
                                          dx

(c2)   diff(sin(x),x,4);

(d2)                                       sin(x)
```

The more general form for differentiating many times with respect to several variables:

```
(c4)   diff (sin(x*y),x,2,y,1);

                                                  2
(d4)                        - 2 y sin(x y) - x y   cos(x y)

We show that this is actually correct:
(c5)   diff (sin(x*y),x);

(d5)                                     y cos(x y)

(c6)   diff (%,x,1);

                                          2
(d6)                                   - y  sin(x y)

(c7)   diff (%,y);

                                                  2
(d7)                        - 2 y sin(x y) - x y   cos(x y)
```



```
(c8)   x^x^x;

                                         x
                                        x
(d8)                                   x

(c9)   diff (%, x);

                         x
                        x    x                         x - 1
(d9)                   x   (x  log(x) (log(x) + 1) + x      )


Note that the character '%' refers to the last displayed item (that is,
the last '(d#)' line.
```

Sometimes it is necessary to tell MAXIMA of dependencies:

```
Suppose we want to tell MAXIMA that the following is the case:
      f = f(x,y)
      g = g(x,y)
      r = r(u,v,w)
      s = s(u,v,w)
      u = u(t).
Then we use the following dependency declaration scheme:

(c11)   depends ([f,g],[x,y],[r,s],[u,v,w],u,t);

(d11)          [f(x, y), g(x, y), r(u, v, w), s(u, v, w), u(t)]

The top-level variable 'DEPENDENCIES' is always set to the list
of functional dependencies currently defined:

(c3)   dependencies;

(d3)           [f(x, y), g(x, y), r(u, v, w), s(u, v, w), u(t)]


Now, to find the derivative of r with respect to t, for example,
```

we can say:

```
(c9)   diff (r,t);

                                           dr du
(d9)                                       -- --
                                           du dt
```

And the derivative of r times s with respect to t is:

```
(c10)   diff (r*s,t);

                                       ds du   dr    du
(d10)                                r -- -- + -- s --
                                       du dt   du    dt
```

The general form of the ‘diff’ function is:

diff (exp, v_1, n_1, v_2, n_2, ...);

which says to differentiate exp n_1 times with respect to v_1, then n_2 times with respect to v_2, and so on.

<u>Integration:</u>

MAXIMA’s ability to perform indefinite and definite integration as helped correct many of the commonly used integration tables. Since the integration and differentiation algorithms work quite independently of each other, we can use the diff function to check the work of the integration function.

The ‘integrate()’ function first tries to *look up* the integrand in a table, and then applies the full Risch integration algorithm. If, in the process of integration, MAXIMA needs to know whether a certain variable is a certain sign, it will ask the user for the information.

The general form of the integrate function is:

integrate (exp, var);

which integrates ‘exp’ with repspect to ‘var’. Examples:


```
(c4)   integrate (cos (3*x), x);

                                   sin(3 x)
(d4)                               --------
                                      3

(c6)   %e ^ (x^2) * x;

                                         2
                                        x
(d6)                                x %e

(c7)   integrate (%, x);

                                        2
                                       x
                                     %e
(d7)                                 ----
                                      2
```

If MAXIMA does not know how to integrate a certain function, then it’ll simply return the integrand still under the integration sign:

```
(c16)   integrate (g(y),y);

                                   /
                                   [
(d16)                              i g(y) dy
                                   ]
                                   /
```

Note that ‘depends’ declarations don’t work with the integrate function. That is, if we’ve said:

```
(c14)   depends (f,x);

(d14)                               [f(x)]
```

```
(c15)   integrate (f, x);

(d15)                                f x
```

Thus, MAXIMA still looks at f as a constant with respect to x.

For definite integration, the general form is:
integrate (exp, var, lo, hi);
where 'lo' and 'hi' are the lower an upper limits, respectively.
The following special names must be used to indicate certain values:

positive infinity	INF
negative infinity	MINF

Examples:

```
(c22)   integrate (x^2-1, x, -10, 10);

                                     1940
(d22)                                ----
                                      3


(c21)   integrate (%e ^ (-x**2), x, MINF, INF);

(d21)                             sqrt(%pi)
```

Here is an example where MAXIMA asks the user some questions about the function being integrated:

```
(c19)    x ** a / (x+1)**(5/2);

                                      a
                                     x
(d19)                             ----------
                                         5/2
                                  (x + 1)
```



```
(c20)   integrate (%, x, 0, INF);
is  a + 1  positive, negative, or zero?

pos;

    2 a + 2
is  -------  an integer?
       5

yes;

is  2 a - 3  positive, negative, or zero?

neg;

                                                  3
(d20)                             beta(a + 1, - - a)
                                                  2

(c23)   x^2 * %e ^ (-u * x^2) * log(x);

                                                2
                                   2   - u x
(d23)                             x  %e        log(x)

(c24)   integrate (d23, x, 0, inf), factor;

is  u  positive, negative, or zero?

pos;

                 sqrt(%pi) (log(u) + 2 log(2) + %gamma - 2)
(d24)          - ------------------------------------------
                                      3/2
                                   8 u
```

Limits:

The facility for taking limits in MAXIMA is a heuristic. There is no known decision procedure for taking limits of arbitrary functions.

The general format of the 'limit' function is:

limit (exp, var, val, dir);

This finds the limit of exp as the real variable var approaches the value of val from the direction dir. Dir may be PLUS or MINUS (for a limit from above and below, respectively) or may be omitted (implying a two-sided limit). Limit these special symbols:

INF	Positive Infinity
MINF	Negative Infinity
UND	Undefined
IND	Indefinite, but bounded
INFINITY	Complex infinity

Examples:

```
(c8)   limit (1 / (x-2), x, 2, minus);

(d8)                                minf

(c10)   limit (1/(x^2-8), x, 2*sqrt(2), plus);

(d10)                                inf

(c11)   limit (sin(x),x,0);

(d11)                                 0

(c12)   limit (tan(x),x, %pi/2);

(d12)                                und

(c13)   limit (cos(x), x, INF);

(d13)                                ind
```


Linear Algebra, Matrices:

We can use MAXIMA to find determinants, inverses, sums, products, eigenvalues, characteristic polynomials etc. of matrices. The operator names are as follows:

operator	explanation
+, -	add and subtract, respectively
'.'	product
^^	inverse
determinant	determinant of the matrix
transpose	transpose of the matrix
eigenvalues	eigenvalues of a matrix - load first
charpoly	characteristic polynomial

Other functions that are available:

```
augcoefmatrix
coefmatrix
copymatrix
diagmatrix
ematrix
entermatrix
genmatrix
matrices
matrix
matrixmap
matrixp
matrix_element_add
matrix_element_mult
matrix_element_transpose
scalarmatrixp
submatrix
zeromatrix
```

Examples:

```
(c29)   m : matrix ([1, 2, 3], [4, 5, 6], [7, 8, 9]);

                                    [ 1  2  3 ]
                                    [         ]
```

```
(d29)                          [ 4  5  6 ]
                               [         ]
                               [ 7  8  9 ]

(c30)   determinant (m);

(d30)                              0

(c31)   m^^-1;
singular

(c32)   m : matrix ([1,2,3],[4,5,6],[7,8,1243]);

                             [ 1  2   3   ]
                             [            ]
(d32)                        [ 4  5   6   ]
                             [            ]
                             [ 7  8  1243 ]

(c33)   m^^-1;

                        [   6167   1231    1    ]
                        [ - ----   ----   ----  ]
                        [   3702   1851   1234  ]
                        [                       ]
                        [  2465     611      1  ]
(d33)                   [  ----   - ----   - --- ]
                        [  1851     1851    617 ]
                        [                       ]
                        [   1        1      1   ]
                        [  ----    - ---   ----  ]
                        [  1234     617    1234  ]
```

To see the results in floating point:

```
(c36)   numer : true;
```



```
(d36)                                true

(c37)   ev (d33);

     [  - 1.665856293895192     0.6650459211237169    0.0008103727714748784  ]
     [                                                                       ]
(d37)[   1.331712587790384     - 0.3300918422474338   - 0.001620745542949757 ]
     [                                                                       ]
     [ 0.0008103727714748784  - 0.001620745542949757  0.0008103727714748784  ]


(c43)   charpoly (m, x);

(d43) ((5 - x) (1243 - x) - 48) (1 - x) + 3 (32 - 7 (5 - x))

                                            - 2 (4 (1243 - x) - 42)


(c44)   solve (%,x);

            sqrt(3) %i   1   sqrt(9310703157346) %i   1906878259 1/3
(d44) [x = (- ---------- - -) (---------------------- + ----------)
                  2        2             3                  27

                   sqrt(3) %i   1
          1537843 (---------- - -)
                       2        2             1249
 + ------------------------------------------ + ----,
     sqrt(9310703157346) %i   1906878259 1/3    3
   9 (---------------------- + ----------)
                3                  27

    sqrt(3) %i   1   sqrt(9310703157346) %i   1906878259 1/3
x = (---------- - -) (---------------------- + ----------)
         2       2             3                  27
```

.
.
.

Matrix products:

```
(c79)   m1 : matrix ([a,b,c], [d,e,f], [g,h,i]);

                                  [ a  b  c ]
                                  [         ]
(d79)                             [ d  e  f ]
                                  [         ]
                                  [ g  h  i ]

(c80)   m1 . m1;

             [              2                                ]
             [ c g + b d + a    c h + b e + a b  c i + b f + a c ]
             [                                               ]
(d80)        [                         2                     ]
             [ f g + d e + a d  f h + e  + b d   f i + e f + c d ]
             [                                               ]
             [                                    2          ]
             [ g i + d h + a g  h i + e h + b g  i  + f h + c g  ]

(c81)   m1 . [1,2,3];

                               [ 3 c + 2 b + a ]
                               [               ]
(d81)                          [ 3 f + 2 e + d ]
                               [               ]
                               [ 3 i + 2 h + g ]
```


Transpose:

```
(c87)   transpose (m1);

                                  [ a  d  g ]
                                  [         ]
(d87)                             [ b  e  h ]
                                  [         ]
                                  [ c  f  i ]
```

Polynomials and Rational Functions: A polynomial is specified using the standard power-basis representation (a univariate poly.):

$a_n * x^n + \ldots + a_0$

A rational function is simply one polynomial divided by another. All the standard operations are defined for polynomials as well. We can perform addition, multiplication, division, integration, differentiation etc.. In addition, MAXIMA supports factoring (of multivariate polynomials), gcd operations, etc..

Examples:

```
(c10)   p (x) := 260*x^6 - 3725*x^5 + 11135*x^4 + 520*x^3 - 7450*x^2+22270*x;

                    6          5            4         3          2
(d10)  p(x) := 260 x  - 3725 x  + 11135 x  + 520 x  - 7450 x  + 22270 x

(c11)   factor (p(x));

                                                          3
(d11)                5 x (4 x - 17) (13 x - 131) (x  + 2)

(c12)   (4*x-17)*(x+1), expand;

                                 2
(d12)                          4 x  - 13 x - 17
```

```
(c13)   gcd (p(x),%);

(d13)                              4 x - 17

(c14)   integrate (p(x),x);

                7         6                            3
           260 x    3725 x          5        4   7450 x           2
(d14)      ------ - ------- + 2227 x  + 130 x  - ------- + 11135 x
             7         6                            3

(c15)   p(x) + (x-1);

              6         5          4        3         2
(d15)    260 x  - 3725 x  + 11135 x  + 520 x  - 7450 x  + 22271 x - 1
```

A multivariate example:

```
(d17) x  y  - 7 y  - 2 x  y  + 11 x  y  + 14 x y  - 77 y  - 22 x  y + x  y

                                                                      2
                                                 + 154 x y - 7 y + 11 x  - 77

(c18)   factor (%);

                          2                   2
(d18)              - (x  - 7) (y + 11) (- y  + 2 x y - 1)
```

Some of the other functions available with factoring:

```
         dontfactor
         expandwrt_factored
         factor
         factorflag
         factorial
         factorout
         factorsum
```



```
         gcfactor
         gfactor
         gfactorsum
         minfactorial
         numfactor
         polyfactor
         savefactors
         scalefactors
         solvefactors
```

Solving polynomial equations:
Consider the problem of solving the following cubic equation:

```
       3       2    2  2        2    2                   3         2
(d28) x  + b x  + a  x  - 9 a x  + a  b x - 2 a b x - 9 a  x + 14 a  x

                                                         3          4
                                                   - 2 a  b  + 14 a  = 0

(c29)   solve (%, x);

                                          2
(d29)                  [x = 7 a - b, x = - a , x = 2 a]
```

This same method can be used to solve systems of equations as well. We simply call ‘solve’ with a list of equations to solve and a list of variables to solve for:

```
        solve ([eq1, eq2, ..., eqn], [var1, var2, ..., varn])
```

This can solve both linear and non-linear systems of equations.
To find out more about solve, do "describe (solve)" in MAXIMA.

Taylor (Laurent) Series:

MAXIMA can produce Taylor series expansions as well. The general form is one of the following:
TAYLOR(exp, var, pt, pow)
TAYLOR(exp,[var_1,pt_1,ord_1],[var_2,pt_2, ord_2],. . .)
TAYLOR(exp, [var_1, var_2, . . .], pt, ord)

The first form does an expansion of exp with respect to the variable var at the point pt upto degree pow. The second gives a truncated power series in the variables var_i about the points pt_i, truncated at ord_i; while the the third is a slightly different syntax for essentially the same operation. See "mhelp (taylor)" for more details.

Examples:

```
(c34)   taylor (sin(x),x,0,5);

                                     3    5
                                    x    x
(d34)/t/                        x - -- + --- + . . .
                                    6    120

(c38)   taylor (cos(x),x,a,3);

                                                 2                  3
                                  cos(a) (x - a)     sin(a) (x - a)
(d38)/t/ cos(a) - sin(a) (x - a) - --------------- + --------------- + . . .
                                         2                  6
```