Introduction to Maxima

Gary Bunting

June 3, 2004

Contents

1 Getting Started 2
2 Obtaining Maxima 5
3 Numbers 7
4 Linear Algebra 10
5 Calculus 13
6 Solving Equations 18
7 Graphics 19
1 Getting Started

Here is a Maxima session:

\begin{verbatim}
turing % maxima
GCL (GNU Common Lisp) Version(2.5.0) Thu Dec 5 08:07:35 EST 2002
Licensed under GNU Library General Public License
Contains Enhancements by W. Schelter
Maxima 5.9.0 http://maxima.sourceforge.net
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.
(C1) f: x^(x-x);
\end{verbatim}

\begin{verbatim}
    x
  (D1) x
(C2) diff(f,x);

    x        x
  (D2) x (x LOG(x) (LOG(x) + 1) + x )
(C3) tex(%);

\$\$x^{x-x}\$, \left(x^{x-x}\right), \log x, \left(\log x+1\right)+x^{x-1} \right)\$
(D3) FALSE
(C4) integrate(D2,x);

    x LOG(x)
  (D4) \%E
(C5) radcan(%);

    x
  (D5) x
(C6) quit();
\end{verbatim}

\begin{verbatim}
turing %
\end{verbatim}
Let us look at this in more detail:

\( \text{(C1)} \) \( f : x^x \cdot x; \)

\[
\begin{align*}
  x \\
  x \\
  x
\end{align*}
\]

\( \text{(D1)} \)

- The first command is the assignment to \( f \) of \( x^x \)
  - The assignment operator in Maxima is the colon :.
  - Statements are terminated a semicolon ;
  - or with a \$ which suppresses printing of the result.
- Maxima input is one-dimensional, output is two-dimensional.
  You can force a one-dimensional display with \texttt{display2d}:

\( \text{(C2)} \) \texttt{display2d : false;} \n
\( \text{(D2)} \) \texttt{FALSE} \\
\( \text{(C3)} \) \texttt{f;} \\
\( \text{(D3)} \) \texttt{x^x^x} \\
\( \text{(C4)} \) \texttt{diff(f,x);} \\
\( \text{(D4)} \) \texttt{x^x^x*(x*x*\texttt{LOG}(x)*(LOG(x)+1)+x^x*(x-1))} \\
\( \text{(C5)} \) \texttt{display2d : true;} \\
\( \text{(D5)} \) \texttt{TRUE} \\
\( \text{(C2)} \) \texttt{diff(f,x);} \\
\( \text{(D2)} \) \\
\( \begin{align*}
  x & x & x - 1 \\
  x & (x \ \texttt{LOG}(x) \ (\texttt{LOG}(x) + 1) + x) \\
\end{align*} \)

\( \text{(C3)} \) \texttt{tex(\%);} \\
\( \texttt{\$x^{x\{x\}}{,},\left(x^x{,},\log x{,},\left(\log x+1\right)+x^x\{x-1\}\right)\right)\$} \\
\( \text{(D3)} \) \texttt{FALSE} \\

- \texttt{diff} is the partial differentiation operator.
- The \% refers to the result of the previous command.
- The \texttt{tex} command converts an expression to \texttt{\LaTeX} which, via cut and paste, can easily be easily included in \texttt{\LaTeX} or \texttt{\LaTeX} documents. The example here results in:

\[
x^x \cdot \left( x^x \ \text{\texttt{LOG}}(x) \ (\text{\texttt{LOG}}(x) + 1) + x^{x-1} \right)
\]
• The FALSE just indicates that the expression has a value which isn’t meaningful to Maxima.

(C4) integrate(D2,x);

\[ x \log(x) \]
\[ \log(x) \] %E

(D4) %E

(C5) radcan(%) ;

\[ x \]

(D5) x

(C6) quit();

• We have used the label D2 to refer to a previous result.

• integrate is the integration operator.

• Maxima doesn’t always give results in the form you expect or want. There are a number of operators like radcan which transform and simplify expressions.

• the command quit() exits maxima.
2 Obtaining Maxima

Maxima for Linux and Windows can be downloaded from the Maxima homepage:

maxima.sourceforge.net

Linux users will also have to download a Common Lisp. The choices are:

- **gcl** GNU Common Lisp
- **clisp** Clisp
- **cmucl** Carnegie-Mellon University Common Lisp

I would recommend **gcl**, which I use and which is used on **turing**, or **clisp** which I have used in the past. The binary of **cmucl** is significantly larger than the others.

User Interfaces

Under Linux there are three user interfaces:

1. **maxima** – this is a plain terminal interface. This is the one I use and is used in all the examples in these notes.

2. **xmaxima** – this is X-windows based interface, but doesn’t seem to offer any advantage over the terminal interface.

3. You can run Maxima under **emacs**. Besides offering the usual **emacs** editing facilities, the **emacsmaxima** package allows Maxima interact with **T\LaTeX** and **B\LaTeX**. To use Maxima under **emacs**, on **turing** go to

   ```
   /usr/share/maxima/5.9.0/doc/EmaximaIntro.ps
   ```

   and

   ```
   /usr/share/maxima/5.9.0/emacs/
   ```

Under windows, unless you are using **emacs**, there is only a **xmaxima**-like interface.
Help

Online help can be obtained with the describe command. You are usually presented with a menu choices.

Suppose we are interested in Bernoulli polynomials.

(C17) describe(bern);

0: BERN :(maxima.info)Definitions for Number Theory.
1: BERNPOLY :Definitions for Number Theory.
2: ZEROBERN :Definitions for Number Theory.
Enter n, all, none, or multiple choices eg 1 3 : 1

Info from file /usr/local/info/maxima.info:
- Function: BERNPOLY (v, n)
  generates the nth Bernoulli polynomial in the variable v.
(D17) FALSE

Try it out:
(C18) bernpoly(z, 10);

\[
\begin{array}{ccccccc}
8 & 2 \\
10 & 9 & 15 z & 6 & 4 & 3 z & 5 \\
z & - 5 z & + & ---- & - 7 z & + & 5 z & - & ---- & + & -- \\
2 & 2 & 66 \\
\end{array}
\]

Errors

Occasionally Maxima gives a cryptic error message and takes you to a Lisp prompt:

(C1)
Error: Console interrupt.
Fast links are on: do (si::use-fast-links nil) for debugging
Error signalled by MACSYMA-TOP-LEVEL.
Broken at SYSTEM::TERMINAL-INTERCEPT. Type :H for Help.
MAXIMA>>

Just type :q to return to Maxima.

MAXIMA>>:q

(C1)
3 Numbers

Integers
Maxima has arbitrary precision integers:

(C21) 100!

(D21) 9332621544394415268169923885626670049071596826438162146859#
2963895217599993229915608941463976156518286253697920827223758251#
185210916864000000000000000000000

And a few number theoretical functions:

(C22) primep(12);

(D22) FALSE

(C23) primep(216091);

(D23) TRUE

(C24) factor(2^63-1);

(D24) 2

7 73 127 337 92737 649657

Exact Arithmetic
Where possible, Maxima uses exact arithmetic:

(C25) sin(%pi/6);

(D25)

1 -
2

(C26) zeta(2);

(D26) 2

%pi
-----
6

(C27) cos(sqrt(2));

(D27) COS(SQRT(2))
Floating Point

The float command converts to floating point:

(C28) \cos(\sqrt{2});

(D28) \cos(SQRT(2))
(C29) float(%);

(D29) 0.15594369476537

Arbitrary precision arithmetic is available:

(C32) fpprec : 100;

(D32) 100
(C33) bfloat(d28);

(D33) 1.55943694765374473454647978908589641624447250391305356890\#

4102677390015211265354586051800302591308267B-1

Complex Numbers

(C34) z1 : 4 + 19*%i;

(D34) 19 %I + 4
(C35) z2 : 3 + 17*%i;

(D35) 17 %I + 3
(C36) z1+z2;

(D36) 36 %I + 7
(C37) z1*z2;

(D37) (17 %I + 3) (19 %I + 4)
(C38) rectform(%);

(D38) 125 %I - 311
(C39) z1/z2;

(D39) 19 %I + 4

(C40) rectform(%);

(D40) 335 11 %I

--- - -----
(C41) polarform(%); 

- \%I \text{ ATAN}(11/335) 
\text{ SQRT}(377) \%E 

(D41) \frac{\text{SQRT}(298)}{298} \frac{\text{SQRT}(298)}{298}
4 Linear Algebra

Matrix Algebra

(C42) a : matrix([a, b, c], [b, c, a], [c, a, b]);

\[
\begin{bmatrix}
    a & b & c \\
    b & c & a \\
    c & a & b
\end{bmatrix}
\]

(D42)

\[
\begin{bmatrix}
    b & c & a \\
    c & a & b \\
    a & b & c
\end{bmatrix}
\]

Matrices are indexed from 1:

(C72) a[2,1];

(D72) b

Matrix multiplication is the \cdot operator. Applied to matrices * does element-by-element multiplication. Similarly matrix powers are found with ^ operator, ^ applied to matrices computes element-by-element powers.

(C43) a*a;

\[
\begin{bmatrix}
    2 & 2 & 2 \\
    a & b & c \\
    \end{bmatrix}
\]

(D43)

\[
\begin{bmatrix}
    2 & 2 & 2 \\
    b & c & a \\
    \end{bmatrix}
\]

(C44) a.a;

\[
\begin{bmatrix}
    2 & 2 & 2 \\
    c + b + a & b c + a c + a b & b c + a c + a b \\
    \end{bmatrix}
\]

(D44)

\[
\begin{bmatrix}
    2 & 2 & 2 \\
    b c + a c + a b & c + b + a & b c + a c + a b \\
    \end{bmatrix}
\]

(b c + a c + a b & b c + a c + a b & c + b + a)
Linear Algebra

Some of the functions available are:

\texttt{determinant(a)}
\texttt{charpoly(a,x)} \quad \text{characteristic polynomial in variable } x
\texttt{eigenvalues(a)}
\texttt{eigenvectors(a)}

(C62) \texttt{charpoly(a,x)};

\begin{equation}
2
\end{equation}

(D62) \((b - x) (C - x) - a)(a - x) + C (a b - C (C - x))

\begin{equation}
b(b - x - a C)
\end{equation}

(C63) \texttt{rat(%, x)};

\begin{equation}
3
\end{equation}

(D63)/R/ \(- x + (C + b + a) x + (C + (- b - a) C + b - a b

\begin{equation}
+ a) x - C + 3 a b C - b - a
\end{equation}

The \texttt{rat} commands puts the expression in \textit{canonical rational} form in the
variable \(x\).

(C69) \texttt{eigenvalues(a)};

\begin{equation}
2
\end{equation}

(D69) \([- \text{SQRT}(C - b C - a C + b - a b + a),

\begin{equation}
2
\end{equation}

\text{SQRT}(C - b C - a C + b - a b + a), C + b + a], [1, 1, 1]]

The list \([1,1,1]\) gives the multiplicities of the eigenvalues.
Generating Matrices

Here is an example of generating Hilbert matrices:

(C40) h[i,j] :- 1/(i + j - 1);

(D40)
\[
\begin{align*}
\mathbf{h} & : \begin{array}{c}
1 \\
i, j & i + j - 1
\end{array}\\
\end{align*}
\]

(C41) h4 : genmatrix(h, 4, 4);

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & - & - \\
2 & 3 & 4 \\
1 & 1 & 1 & 1 \\
- & - & - & - \\
2 & 3 & 4 & 5 \\
\end{bmatrix}
\]

(D41)

(C42) h4**(-1);

\[
\begin{bmatrix}
16 & -120 & 240 & -140 \\
-120 & 1200 & -2700 & 1680 \\
240 & -2700 & 6480 & -4200 \\
-140 & 1680 & -4200 & 2800 \\
\end{bmatrix}
\]

The :- is the assignment operator for functions (more later).
5 Calculus

Differentiation
diff is the partial differentiation operator.

(C6) \[ g : x^2 \cdot \sin(y) + a \cdot \log(x^2)/y; \]

\[
\frac{a \ \log(x \ z)}{y} - \frac{x \ \sin(y) \ z}{y}
\]

Here is the partial derivative \( \partial^3 g / \partial x \partial y^2 \):

(C7) \[ \text{diff}(g, x, 1, y, 2); \]

\[
\frac{2 \ \log(x \ z)}{y} - \frac{x \ \sin(y) \ z}{y}
\]

If you use diff without any variables it gives the total derivative:

(C8) \[ \text{diff}(g); \]

\[
\frac{a \ \log(x \ z)}{y} - \frac{x \ \sin(y) \ z}{y}
\]

(D8) \[
\left( \frac{-x \ \sin(y)}{y} \right) \ \text{DEL}(z) + \left( \frac{x \ \cos(y)}{y} \right) \ \text{DEL}(y)
\]

\[
+ \left( \frac{-x \ \sin(y)}{y} \right) \ \text{DEL}(x) + \left( \frac{-x \ \sin(y)}{y} \right) \ \text{DEL}(a)
\]

Dependence of Variables

In the Taylor series method for the solution of ODEs we start with a differential equation

\[
\frac{dy}{dt} = f(t, y)
\]

and differentiate to get successive terms of the Taylor series

\[
y(t + \Delta t) = y(t) + \frac{dy}{dt} \Delta t + \frac{1}{2} \frac{d^2 y}{dt^2} \Delta t^2 + \frac{1}{6} \frac{d^3 y}{dt^3} \Delta t^3 + \ldots
\]

First we specify that \( f \) is a function of \( y \) and \( t \) and that \( y \) is a function of \( t \):

(C235) \[ \text{depends}(f, [y, t], [y, t]); \]

(D235) \[ [f(y, t), y(t)] \]
Now we can take our differential equation and differentiate:

(C291) \( \text{de} : \text{diff}(y,t) - f; \)

\[
\frac{dy}{dt} -- - f
\]

(D291)

(C292) \( \text{de1} : \text{diff}(\text{de},t); \)

\[
\frac{2}{d^2y + df dy + df}
\]

(D292)

(C297) \( \text{de2} : \text{diff}(\text{de},t,2); \)

\[
\frac{3}{d^3y + df dy + df}
\]

(D297)

The next thing to do is to substitute for the derivatives of \( y \).

**Substitution**

The command

\[ \text{subst}([\text{var}_1 = \text{expr}_1,\ldots], \text{expr}) \]

substitutes expressions for variables in an expression.

For our example:

(C293) \( \text{de1} : \text{subst}(\text{de},\text{de1}); \)

\[
\frac{2}{d^2y + df dy + df}
\]

(D293)

(C298) \( \text{de2} : \text{subst}([\text{de},\text{de1}],\text{de2}); \)

\[
\frac{3}{d^3y + df dy + df}
\]

(D298)
Integration

While differentiation is a mechanical procedure, integration in closed form may or may not be possible.

There is an algorithm, the Risch algorithm, which can decide whether a given function has an integral in closed form and, if so, computes the integral. Maxima, and other computer algebra systems, have an incomplete implementation of the Risch algorithm, integrals are usually computed by various ad-hoc techniques.

Here are some examples:

(C92) integrate(x^n,x);
Is n + 1 zero or nonzero?

nonzero;

n + 1
x

(D92) ------
n + 1

(C101) assume(a>0);

(D101) [a > 0]

(C102) assume(b>0);

(D102) [b > 0]

(C103) integrate(1/(a+b*x^2),x);

SQRT(b) x

ATAN(--------)

SQRT(a)

(D103) ------------------

SQRT(a) SQRT(b)

Definite Integrals

(C104) integrate(a*exp(-x^2), x, 0, inf);

SQRT(%PI) a

(D104) --------------

2

Change of Variables

The command

changevar(expr, f(x,y), y, x)
makes the change of variable given by \( f(x, y) - 0 \) in the integral `expr` with respect to \( x \) to the new variable \( y \).

(C31) \( f : \log(x)^3/x; \)

\[
\frac{3}{\log(x)} \quad \frac{\text{---}}{x}
\]

(D31)

(C32) `intf : 'integrate(f, x);`

\[
/ \quad 3
\]

\[
[ \log(x)
\]

\[
] \quad x
\]

(D32)

(C33) `inty : changevar(intf,x - exp(y),y,x);`

\[
/ \quad 3
\]

\[
] \quad y \quad dy
\]

(D33)

(C34) `ev(y,integrate);`

\[
4 \quad y
\]

(D34)

\[
4
\]

- The quote `'` prevents evaluation of an expression.
- Then we need `ev` to force evaluation.

**Taylor Series**

The command

\[
taylor(expr, x, x0, n)
\]

computes the Taylor series of `expr` in the variable `x` at the point `x0` to order `n`.

(C134) \( f : (1 - \sqrt{1 - e^2})/e; \)

\[
\frac{2}{1 - \sqrt{1 - e}} \quad \frac{\text{---}}{e}
\]

(D134)
(C135) taylor(f, e, 0, 10);

\[
\begin{array}{cccc}
3 & 5 & 7 & 9 \\
e & e & e & e \end{array}
\]

(D135)/T/
- + -- + -- + ---- + ---- + ... 
2 8 16 128 256

**Limits**

The command

\[ \text{limit} \left( \text{expr} \text{, } x \text{, } x_0 \right) \]

computes the limit \( x \to x_0 \) of the expression \( \text{expr} \).

(C36) \( f : x \text{log}(x) \);

(D36) \( x \text{ LOG}(x) \)

(C37) \( \text{limit} \left( f \text{, } x \text{, } 0 \right) \);

(D37) \( 0 \)

(C38) \( f : \text{sin}(3\text{x})^2/(1 - \cos(x)) \);

(D38) \[
\frac{2}{\text{SIN} \left( 3 \text{x} \right)}
\]

(D39) \[
\frac{\text{-----------}}{1 - \text{COS}(x)}
\]

(C39) \( \text{limit} \left( f \text{, } x \text{, } 0 \right) \);

(D39) \( 18 \)
6 Solving Equations

The command `solve` can be used to solve a single equation:

```
solve(expr, var)
```

or a system of equations:

```
solve([expr_1, .. , expr_n],[var_1, .. ,var_k])
```

If any of the expressions is not an equation, it is set equal to zero to make an
equation. `solve` returns a list of solutions.

(C33) `solve(x*sin(x) - x, x);

SOLVE is using arc-trig functions to get a solution.
Some solutions will be lost.

```
2 %PI
[x = --, x = 0]
2
```

Maxima tries to find exact solutions, but will give numerical solutions if it
has difficulty.

Here is an example involving two equations in two unknowns.

(C5) `e1 : 3*x^2 + 2*x*y + 5*y;

```
2
2
x y + 5 y + 3 x
```

(C6) `e2 : x^2 + y^2 - 1;

```
2
2
y + 2 x - 1
```

(C7) `solve([e1=0,e2=0],[x,y]);

```
[[x = 0.92666317443688, y = 0.37589255355321],
 [x = 0.81129132875858, y = 0.58464215977587],
 [x = 1.369011743157652 %I - 0.82691675689563,
  y = 0.69276628856866 %I + 1.634113509106991],
 [x = 1.369011743157652 %I - 0.82691675689562,
  y = 1.634113509106991 - 0.69276628856866 %I]]

18
7 Graphics

Maxima has pretty basic 2D and 3D plotting facilities via Gnuplot.

```
(C20) plot2d([sin(x),cos(x),sin(x)*cos(x)],[x,-10,10]);
```

![Graph of sin(x), cos(x), and sin(x)cos(x)](image-url)