Maxima and Emacs

Jay Belanger

1 Using Maxima and Emacs

Emacs, while nominally a text editor, is an extensible environment for interacting with text in various ways. As such, it provides a convenient interface for many text-based utilities. For example, users of Emacs can interact with *Maxima* from within their favorite editor[1] in a number of ways. The most basic way is to use Emacs as a shell and run a *Maxima* process interactively from within a buffer. Emacs can interact with this process from other buffers, and thus interact with the *Maxima* process in less straightforward ways. Emacs can also be used to write *Maxima* programs. A special mode is provided which will take care of syntax highlighting, indentation, etc. Portions of the program can also be sent automatically to a *Maxima* program. Finally, text and *Maxima* commands can be interweaved in a *Maxima* notebook mode called *EMaxima*.

2 Running Maxima Interactively

To run *Maxima* interactively in a buffer, type `M-x maxima`. A buffer named `*Maxima*` should start up in which a *Maxima* process is running.[2] A history of input commands can be saved across Maxima sessions by setting the variable `maxima-save-input-history` to `t`. In the `*Maxima*` buffer, `RET` will check the line for balanced parentheses, and send line as input. `C-RET` will send the line as input without checking for balanced parentheses. A list of available commands are listed in Appendix C.

If this is the only way that *Maxima* is to be used from within Emacs, then the *Imaxima* package should be considered. It allows the user to work with *Maxima* interactively, and produces the output in visually nice form. It is available from `http://memberss3.jcom.home.ne.jp/imaxima/Site/Welcome.html`.

[1] Emacs.

[2] In XEmacs, a double input prompt will sometimes appear in the `*Maxima*` buffer. If this occurs, the customizable variable `maxima-fix-double-prompt` can be set to `t` to take care of this.

3 Running Maxima from arbitrary buffers

Several commands will access the *Maxima* process from outside of the `*Maxima*` buffer; these commands are given convenient key bindings with the *Maxima* minor mode (which can be started with `M-x maxima-minor-mode`), but are available without the minor mode. In the following, the minor mode keybindings are given in parentheses.

The command `M-x maxima-minibuffer-on-determined-region` (`C-c = e`) will send part of the current buffer containing the point to the *Maxima* process and return the result in the minibuffer. The region that is sent is the region bounded above and below by blank lines (although the delimiters can be changed by changing the regexps `maxima-minor-prefix` and `maxima-minor-postfix`). Given an argument, this command will also insert the output into the current buffer, after the symbol `==>` . (The output indicator `==>` is the value of the customizable variable `maxima-minor-output`.) When calling `maxima-minibuffer-on-determined-region` on a region which already has some output displayed, everything after the indicator `==>` will be ignored, and, if the new output is to be inserted, everything after the indicator will be assumed to be old output and deleted.

The commands `M-x maxima-minibuffer-on-region` (`C-c = r`), `M-x maxima-minibuffer-on-line` (`C-c = l`) and `M-x maxima-minibuffer-on-form` (`C-c = f`) work similarly to `M-x maxima-minibuffer-on-determined-region`, but send the current region (respectively, the current line, current form[3]) to *Maxima* and display the result in the minibuffer. Care must be taken when inserting the output into the current buffer with `M-x maxima-minibuffer-on-region` or `M-x maxima-minibuffer-on-form`, since new output will be inserted without the old output being deleted.

The command `M-x maxima-minibuffer` (`C-c = m`) will use the minibuffer to prompt for an expression to send to a *Maxima* process, and the result will appear in the minibuffer[4]. You can scroll through previous minibuffer inputs using the arrow keys.

The latest *Maxima* output (whether from a minibuffer command or not) can be placed in the current buffer with the command `M-x maxima-insert-last-output` (`C-c = o`).[5] The command `M-x maxima-insert-last-output-tex` (`C-c = t`) will insert the last output in TeX form.

4 Maxima mode

Maxima mode is a major mode for writing *Maxima* code. While in this mode, Emacs provides commands for moving around in the code, syntax highlighting, indenting lines to an appropriate level, sending portions of the code to a *Maxima*

[3] The region between the preceding `;` or `$` and the subsequent `;` or `$`.

[4] In GNU Emacs, the output will be in 2D form if the customizable variable `maxima-minibuffer-2d` is non-nil. Since XEmacs does not have a resizable minibuffer, this cannot be done in XEmacs.

[5] If the output is 2D, this won't look nice.

process, and providing help on *Maxima* commands. A list of available commands is in Appendix E.

When something is sent to a *Maxima* process, the `*Maxima*` buffer will appear. (If a *Maxima* process is not running, one will be started.) It can also be made to appear by using the command `C-c C-p`. If an argument is given to a command to send information to *Maxima*, the region will first be checked to make sure the parentheses are balanced. The Maxima process can be killed, after asking for confirmation, with `C-c C-k`. To kill without confirmation, give `C-c C-k` an argument.

By default, a newline will be indented to the same level as the previous line, with an additional space added for open parentheses. A tab will add extra spaces, by default, this is 2. The behaviour of the automatic indentation can be changed by the command `M-x maxima-change-indent-style`. The possibilities are `'standard`, as above, and `'perhaps-smart`, which tries to guess an appropriate indentation, based on open parentheses, “do” loops, etc.

A `RET` will, by default, insert a new line, and indent the new line an appropriate amount. This behavior can be changed by setting the value of `maxima-return-style`, the self-explanatory options are `'newline`, `'newline-and-indent`, and `'reindent-then-newline-and-indent`. (Note that standard Emacs behavior is to let `RET` only insert a newline, and `LFD` or `C-j` insert a newline and indent it.)

The indentation style, as well as many other things, are customizable; see Appendix H.1.

To help writing *Maxima* code in `Noweb`, `maxima-noweb-mode` is a modification of `maxima-mode` that will work nicely with `noweb-mode`; namely, it will limit any relevant searches to the current chunk and treat `<<...>>` as word parts.

5 EMaxima

EMaxima is a major mode for Emacs that allows the user to write documents while interacting with *Maxima*. It is based on Dan Dill's TEX/*Mathematica* package[6], and uses a modified version of William Schelter's `maxima.el`. While the *Maxima* mode provided by `maxima.el` is designed to help write *Maxima* programs, *EMaxima* is designed to help write documents that include *Maxima* code. *EMaxima* is an extension of the LATEX mode provided by AUCTEX[7], and so has the LATEX mode commands available. The resulting document can be processed by LATEX; this requires putting

```
\usepackage{emaxima}
```

in the preamble.

[6] TEX/*Mathematica* is available from `ftp://chem.bu.edu/pub/tex-mathematica-2.0`.

[7] This can be configured so that *EMaxima* extends the standard TEX mode provided by Emacs, or just text mode.

5.1 Cells

The basic unit of *Maxima* code in *EMaxima* is a **cell**. A cell consists of text between the delimiters

```
\begin{maxima}
```

and

```
\end{maxima}
```

A cell can be created by typing `C-c C-o`. (The `C-o` in this case stands for **o**pening a cell.) The delimiters will then be placed in the buffer, and the point will be placed between them.

When working with several cells, you can jump between them by using `C-c +` to go to the next cell and `C-c -` to go to the previous cell.

5.2 Evaluating cells

To evaluate the contents of a cell, the command `C-c C-u c` (`emaxima-update-cell`)[8] will send the contents of the cell to a *Maxima* process (if there is no *Maxima* process running, one will be started) and return the results to the cell, separated from the input by the marker

```
\maximaoutput
```

To differentiate $\sin(x^2)$, for example, type `diff(sin(x^2),x);` in a cell:

```
\begin{maxima}
diff(sin(x^2),x);
\end{maxima}
```

After typing `C-c C-u c`, it will look like

```
\begin{maxima}
diff(sin(x^2),x);
\maximaoutput
                                                   2
                                        2 x cos(x )
\end{maxima}
```

To delete the output and return the cell to its original form, you can use the command `C-c C-d`. If the document is to be TeXed, the above cell will look like:

[8]Sending the cells contents to a *Maxima* process and returning the results is called **updating** the cell, the prefix `C-c C-u` will be used to update cells in different ways.

Maxima

```
diff(sin(x^2),x);
```

and the cell with output will look like:

Maxima

```
diff(sin(x^2),x);
```

```
2
                                  2 x cos(x )
```

EMaxima mode can take advantage of the fact that *Maxima* can give its output in LaTeX form. The command `C-c C-u C` works the same as `C-c C-u c`, except now the output is in LaTeX form, ready to be formatted by LaTeX. In general, if `C-c C-u` *letter* returns *Maxima* output, then `C-c C-u` *capital letter* will return the output in TeX form. The above cell would become

```
\begin{maxima}
diff(sin(x^2),x);
\maximaoutput*
\m  2\,x\,\cos x^2 \\
\end{maxima}
```

which, when LaTeXed, would become

Maxima

```
diff(sin(x^2),x);
```

$$2\,x\,\cos x^2$$

(Note that whenever a cell is updated, any old output is discarded and replaced with new output.) The command `C-c C-u a` will update all of the cells in your document, stopping at each one to ask if you indeed want it updated. Given an argument, `C-u C-c C-u a`, it will update all of the cells in your document without asking. The command `C-c C-u A` behaves similarly, except now all the output is returned in LaTeX form.

5.3 Referencing Other Cells

Instead of *Maxima* code, a cell can contain a reference to another cell, and when the original cell is sent to *Maxima*, the reference is replaced by the referenced cell's contents (but only in the *Maxima* process buffer, the cell's content in the document's buffer is not changed). In order to do this, the original cell must be marked by having a label of the form <*filename*:*cell label*>. (The reason for the *filename* will become apparent later, and *cell label* is optional for the referencing cell.) The referenced cell must also be labeled, with the same *filename* but a unique *cell label*. To reference the other cell, the original cell need only contain the marker for the referenced cell. For example, given cell 1:

```
\begin{maxima}[filename:optional]
<filename:definef>
diff(f(x),x);
\end{maxima}
```

and cell 2:

```
\begin{maxima}[filename:definef]
f(x):=sin(x^2);
\end{maxima}
```

then the result of updating cell 1 (`C-c C-u c`) will be:

```
\begin{maxima}[filename:optional]
<filename:definef>
diff(f(x),x);
\maximaoutput

                                                   2
                                       f(x) := sin(x )

                                                 2
                                         2 x cos(x )
\end{maxima}
```

When LaTeXed, the top line will contain a copy of the marker.

Maxima — filename:optional

```
<filename:definef>
diff(f(x),x);
```

```
                                   2
                       f(x) := sin(x )
```

```
                                   2
                          2 x cos(x )
```

A cell can contain more than one reference, and referenced cells can themselves contain references.

To aid in labelling the cells, the command `C-c C-x` will prompt for a label name and label the cell. To aid in calling references, the command `C-c C-TAB` can be used for completing the the *filename* and *cell label* parts of a reference, based on the current labels. Another option is to set the Emacs variable `emaxima-abbreviations-allowed` to `t`, say, by putting the line

```
(setq emaxima-abbreviations-allowed t)
```

in your `.emacs` file. This will allow the *filename* and *cell label* parts of a reference to be abbreviated by enough of a prefix to uniquely identify it, followed by ellipses `...` For example, if there are cells labelled

```
[filename:long description]
[filename:lengthy description]
```

Then the reference

```
<...:le...>
```

will suffice to refer to the second label above.

If you want the references in a cell to be replaced by the actual code, the command `C-c @` will expand all the references and put the code into a separate buffer (so it will not affect the original document).

5.4 WEB

The reason for the ability to reference other cells is so that you can write what Donald Knuth calls literate programs. The idea is that the program is written in a form natural to the author rather than natural to the computer. (Another aspect of Knuth's system is that the code is carefully documented, hence the name "literate programming", but that is done naturally in *EMaxima*.) Knuth called his original literate programming tool `WEB`, since, as he puts it, "the structure of a software program may be thought of as a web that is made up of many interconnected pieces." *EMaxima*'s ability in this respect is taken directly from TEX/*Mathematica*, and is ultimately based on `WEB`. To create a program, the "base cell" or "package cell" should contain a label of the form [*filename*:] (no cell label), and can contain references of the form <*filename*:*part*> (same file name as the base cell).

As a simple (and rather silly) example, suppose we want to create a program to sum the first n squares. We could start:

```
\begin{maxima}[squaresum.max:]
squaresum(n) := (
  <squaresum.max:makelist>
  <squaresum.max:squarelist>
  <squaresum.max:addlist>
  );
\end{maxima}
```

We would then need cells

```
\begin{maxima}[squaresum.max:makelist],
L:makelist(k,k,1,n),
\end{maxima}

\begin{maxima}[squaresum.max:squarelist]
<squaresum.max:definesquare>
L:map(square,L),
\end{maxima}

\begin{maxima}[squaresum.max:addlist]
lsum(k,k,L)
\end{maxima}
```

and then we would also need:

```
\begin{maxima}[squaresum.max:definesquare]
square(k) := k^2,
\end{maxima}
```

When TEXed, the header of the cell will say that it determines the file `squaresum.mu`.

Maxima — Definition of squaresum.max

```
squaresum(n) := (
  <squaresum.max:makelist>
  <squaresum.max:squarelist>
  <squaresum.max:addlist>
  );
```

The command `C-u C-c @` will put all the pieces together in the file it determines. The resulting file, in this case, will be `squaresum.max` and will look like:

```
squaresum(n) := (
  L:makelist(k,k,1,n),
  square(k) := k^2,
  L:map(square,L),
```

```
lsum(k,k,L)
);
```

(Although the idea is that only the computer need look at this file.)

5.5 Other types of cells

When a cell is TEXed, the input and output are kept separate. To have the results look like a *Maxima* session, there are, in addition to the standard cells, special cells called *session cells*. A session cell is delimited by

```
\begin{maximasession}
```

and

```
\end{maximasession}
```

The command `C-c C-a` will create a session cell. When a session cell is updated, the portion of the cell after the `\maximaoutput` will contain both the input and the output, with the *Maxima* prompts. For example, if the session cell

```
\begin{maximasession}
diff(sin(x),x);
integrate(cos(x),x);
\end{maximasession}
```

were updated, the result would look like

```
\begin{maximasession}
diff(sin(x),x);
integrate(cos(x),x);
\maximaoutput
(%i1) diff(sin(x),x);


(%o1)                                   cos(x)
(%i2) integrate(cos(x),x);


(%o2)                                   sin(x)
\end{maximasession}
```

which, when TEXed, would look like

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


(%o1)                                   cos(x)
(%i2) integrate(cos(x),x);


(%o2)                                   sin(x)
```

If it is updated in TEX form, it will look like

```
\begin{maximasession}
diff(sin(x),x);
integrate(cos(x),x);
\maximaoutput*
\i5.  diff(sin(x),x); \\
\o5.  \cos x \\
\i6.  integrate(cos(x),x); \\
\o6.  \sin x \\
\end{maximasession}
```

which, when TEXed, will look like

```
(%i5)  diff(sin(x),x);
(%o5)
```

$$\cos x$$

```
(%i6)  integrate(cos(x),x);
(%o6)
```

$$\sin x$$

If a cell is *starred*, that is, if the environment ends with an asterisk, then the output will not appear in the TEXed output. The star can be toggled with `C-c C-n`.

If the command to create one type of cell is called while inside another type of cell, the type of cell will be changed. So, for example, the command `C-c C-a` from inside the cell

```
\begin{maxima}
diff(x*sin(x),x);
\end{maxima}
```

will result in

```
\begin{maximasession}
diff(x*sin(x),x);
\end{maximasession}
```

If a standard cell is a package part, its type cannot be changed.

5.6 EMaxima and `Preview-LaTeX`

For users of `preview-latex`, the *EMaxima* cells can be previewed.[9] This requires using the `preview` option to the `emaxima` package; i.e., putting

[9]Currently, this only works in GNU Emacs.

```
\usepackage[preview]{emaxima}
```

in the preamble. The cells will be un-previewed whenever they are updated. If the customizable variable `emaxima-preview-after-update-all` is non-nil, then the buffer will be re-previewed whenever an update-all command is called.

5.7 Miscellaneous

Some *Maxima* commands can be used even outside of cells. The command `C-c C-u l` send the current line to a *Maxima* process, comment out the current line, and insert the *Maxima* output in the current buffer. The command `C-c C-u L` will do the same, but return the result in LaTeX form.

The command `C-c C-h` will provide information on a prompted for function (like *Maxima*'s `describe`), and `C-c C-i` will give the *Maxima* info manual.

Finally, the *Maxima* process can be killed with `C-c C-k`.

A list of the commands for *EMaxima* are in Appendix F, and the customizability options are in Appendix H.2.

A Installation

To run *Maxima* interactively or use *Maxima* mode or minor mode, the files `maxima.el` and `maxima-font-lock.el` need to be somewhere in the Emacs load path. To be able to automatically run *Maxima* or use *Maxima* mode or minor mode, add the lines

```
(autoload 'maxima "maxima" "Run Maxima interactively" t)
(autoload 'maxima-mode "maxima" "Major mode for writing Maxima programs" t)
(autoload 'maxima-minor-mode "maxima" "Minor mode for working with Maxima" t)
```

in your `.emacs` file. To ensure that files ending in `.max` start up in *Maxima* mode, add the line

```
(setq auto-mode-alist (cons '("\\.max" . maxima-mode) auto-mode-alist))
```

to `.emacs`.

For the *EMaxima* package, in addition to the above files, the files `emaxima.el` and `emaxima.lisp` need to be somewhere in the Emacs load path[10], and if you want to run LaTeX on the resulting document, `emaxima.sty` and `maxima.sty` need to be in the TeX inputs path. If you use pdflatex, you'll also need `pdfcolmk.sty`.

To make sure that `emaxima.el` is loaded when necessary, the line

```
(autoload 'emaxima-mode "emaxima" "EMaxima" t)
```

can be inserted into your `.emacs` file. Then typing `M-x emaxima-mode` will start *EMaxima* mode. The command `M-x emaxima-mark-file-as-emaxima` will put the line

```
%-*-EMaxima-*-
```

at the beginning of the file, if it isn't there already, and will ensure that the next time the file is opened, it will be in `emaxima-mode`. This can be done automatically everytime a file is put in `emaxima-mode` by putting the line

```
(add-hook 'emaxima-mode-hook 'emaxima-mark-file-as-emaxima)
```

somewhere in your `.emacs` file.

B Maxima help commands

The following commands can be used in any Maxima related buffer. (For Maxima minor mode, replace `C-c C-d` with `C-= d`.)

[10] If Emacs cannot find `emaxima.lisp`, then the TeX output functions will not work, any attempts to get TeX output will result in an error.

Key	Description
`C-c C-d h`	Get help on a (prompted for) subject.
`C-c C-d d`	
`C-c C-d C-d`	Get help with the symbol under point.
`C-c C-d a`	
`C-c C-d a`	Apropos.
`C-c C-d p`	
`C-c C-d C-p`	Get apropos with the symbol under point.
`C-c C-d m`	
`C-c C-d C-m`	
`C-c C-d i`	
`C-c C-d C-i`	Read the Maxima info manual.

C Interactive Maxima commands

Key	Description
`M-TAB`	Complete the Maxima symbol as much as possible, providing a completion buffer if there is more than one possible completion.
`C-M-TAB`	Complete the input line, based on previous input lines.
`C-c C-k`	Kill the process and the buffer, after asking for confirmation. To kill without confirmation, give `C-c C-k` an argument.
`M-p`	Bring the previous input to the current prompt.
`M-n`	Bring the next input to the prompt.
`M-r`	Bring the previous input matching a regular expression to the prompt.
`M-s`	Bring the next input matching a regular expression to the prompt.

D Maxima minor mode commands

Key	Description
`C-c=e`	Run Maxima on the region between `maxima-minor-prefix` and `maxima-minor-postfix`. By default, these are blank lines.
`C-c=r`	Run Maxima on the current region.
`C-c=l`	Run Maxima on the current line.
`C-c=f`	Run Maxima on the current form.
`C-c=m`	Prompt for Maxima input in the minibuffer.
`C-c=o`	Insert the last Maxima output in the current buffer.
`C-c=t`	Insert the last Maxima output in TeX form in the current buffer.

E Maxima mode commands

Motion

Key	Description
`M-C-a`	Go to the beginning of the form.
`M-C-e`	Go to the end of the form.
`M-C-b`	Go to the beginning of the sexp.
`M-C-f`	Go to the end of the sexp.

Process

Key	Description
`C-c C-p`	Start a *Maxima* process.
`C-c C-r`	Send the region to the *Maxima* process.
`C-c C-b`	Send the buffer to the *Maxima* process.
`C-c C-c`	Send the line to the *Maxima* process.
`C-c C-e`	Send the form to the *Maxima* process.
`C-c C-k`	Kill the *Maxima* process.
`C-c C-p`	Display the *Maxima* buffer.

Completion

Key	Description
`M-TAB`	Complete the *Maxima* symbol.

Comments

Key	Description
`C-c ;`	Comment the region.
`C-c :`	Uncomment the region.
`M-;`	Insert a short comment.
`C-c *`	Insert a comment environment.

Indentation

Key	Description
`TAB`	Indent line.
`M-C-q`	Indent form.

Miscellaneous

Key	Description
`M-h`	Mark the form.
`C-c )`	Check the region for balanced parentheses.
`C-c C-)`	Check the form for balanced parentheses.

F EMaxima mode commands

Key	Description
`C-c C-o`	Create a (standard) cell.
`C-c C-a`	Create a session cell.
`C-c C-n`	Toggle starred cells.
`C-c +`	Go the the next cell.
`C-c -`	Go to the previous cell.
`C-c C-u a`	Update all of the cells. With an argument, don't ask before updating.
`C-c C-u A`	Update all of the cells in TEX form. With an argument don't ask before updating.
`C-c C-u s`	Update all of the session cells in TEX form. With an argument, don't ask before updating.

Commands only available in cells.

Key	Description
`C-c C-v`	Send the current cell to the *Maxima* process.
`C-c C-u c`	Update the current cell.
`C-c C-u C`	Update the current cell in TEX form.
`C-c C-d`	Delete the output from the current cell.
`C-c C-x`	Insert a heading for the cell indicating that it's part of a package.
`C-c @`	Assemble the references contained in the cell. With an argument, assemble the package that the cell defines.
`C-c C-TAB`	Complete a reference within a cell.

Commands only available outside of cells.

Key	Description
`C-c C-u l`	Send the current line to *Maxima*, and replace the line with the *Maxima* output.
`C-c C-u L`	Send the current line to *Maxima*, and replace the line with the *Maxima* output in TEX form.

G AUCTEX commands

Inserting commands

Key	Description
`C-c C-e`	Insert an environment.
`C-c C-s`	Insert a section.
`C-c ]`	Close an environment.
`C-c C-j`	Insert an item into a list.
`"`	Smart quote.
`$`	Smart dollar sign.
`C-c @`	Insert double brace.
`C-c C-m`	Insert TEX macro.
`M-TAB`	Complete TEX macro.

Formatting

Key	Description
`C-c C-q C-r`	Format region.
`C-c C-q C-s`	Format section.
`C-c C-q C-e`	Format environment.
`C-c .`	Mark an environment.
`C-c *`	Mark a section.

Commenting

Key	Description
`C-c ;`	Comment a region.
`C-u C-c ;`	Uncomment a region.
`C-c %`	Comment a paragraph.
`C-u C-c %`	Uncomment a paragraph.

Font selection

Key	Description
`C-c C-f C-b`	Bold.
`C-c C-f C-i`	Italics.
`C-c C-f C-r`	Roman.
`C-c C-f C-e`	Emphasized.
`C-c C-f C-t`	Typewriter.
`C-c C-f C-s`	Slanted.
`C-c C-f C-d`	Delete font.
`C-u C-c C-f`	Change font.

Running TEX

(Commands: `TeX`, `TeX Interactive`, `LaTeX`, `LaTeX Interactive`, `SliTeX`, `View`, `Print`, `BibTeX`, `Index`, `Check`, `File`, `Spell`.)

Key	Description
`C-c C-c`	Run a command on the master file.
`C-c C-r`	Run a command on the current region.
`C-c C-b`	Run a command on the buffer.
`` C-c ` ``	Go to the next error.
`C-c C-k`	Kill the TEX process.
`C-c C-l`	Center the output buffer.
`C-c C-^`	Switch to the master file.
`C-c C-w`	Toggle debug of overful boxes.

H Customization

H.1 Customizing Maxima mode

Indentation in *Maxima* mode can be controlled with the following variables:

maxima-indent-style This determines how hard *Maxima* mode will try to compute an appropriate indentation. The options are `'standard` and `'perhaps-smart`, the default is `'perhaps-smart`.

maxima-indent-amount This is the default indentation for each level in *Maxima* mode. By default, it is 2.

maxima-paren-indent-amount This is the amount of extra indentation to be added for each parenthetical nesting. By default, it is 1, so that the lines following an open parenthesis will start in the first column past the parenthesis.

maxima-blockparen-indent-amount This is the amount of extra indentation to be added after a `block(`. By default, this is -3, so that the following lines won't be indented too far.

maxima-continuation-indent-amount This is the amount of indentation to be added when a line is the continuation of the expression begun on the previous line. By default, it is 2.

maxima-multiline-comment-indent-amount This is the amount of extra indentation to be given to comments. By default, it is 2.

maxima-if-extra-indent-amount This is the amount of extra indentation to give a then which follows an if, assuming the then starts on its own line. By default, it is 0, so the then will begin right under the if.

Other facets of *Maxima* mode can be controlled through the following options:

maxima-return-style This determines how *Maxima* mode will handle `RET`. The options are `'newline`, `'newline-and-indent`, and `'reindent-then-newline-and-indent`, the default is `'reindent-then-newline-and-indent`.

Some other options that may occasionally need to be set are:

maxima-command The command used to start *Maxima*. By default, it is `maxima`.

maxima-args Extra arguments to pass to the *Maxima* command. By default, it is `nil`.

maxima-use-tabs If this is non-nil, the indentation will use TABS as well as spaces. By default, it is `nil`.

H.2 Customizing EMaxima

There are a few (very few) things that you can do to customize *EMaxima*.

By default, *EMaxima* is an extension of AUCTEX mode. This can be changed by changing the variable `emaxima-use-tex`. The possible values are `'auctex`, `'tex` and `nil`. Setting `emaxima-use-tex` (the default) to `'auctex` will make *EMaxima* an extension of AUCTEX, setting it to `'tex` will make *EMaxima* an extension of Emacs's default TEX mode, and setting `emaxima-use-tex` to `nil` will make *EMaxima* an extension of text-mode. So, for example, putting

```
(setq emaxima-use-tex nil)
```

in your `.emacs` file will make *EMaxima* default to an extension of text mode.

Whether or not the dots (...) abbreviation is allowed in cell references is controlled by the elisp variable `emaxima-abbreviations-allowed`, which is set to `t` by default. Setting this to `nil` will disallow the abbreviations, but will speed up package assembly.

The variable `emaxima-preview-after-update-all` will determine whether or not the buffer will be previewed (when preview-latex is being used) after an update-all command. By default, it is `t`.

The LaTeXed output can be customized somewhat. The `emaxima` LaTeX package can take some options, namely `breqn`, `lines`, `listings` and `preview`. The `breqn` option will use the LaTeX `breqn` package (which must be installed) to break long *Maxima* lines into shorter lines automatically. The `preview` option will enable the Emacs preview package (which must be installed) to preview the *Maxima* environments. The `lines` option will put lines before and after some of the environments. The `listings` option will use the LaTeX `listings` package (which must be installed) to typeset some of the *Maxima* code. To use this, the file `maxima.sty` must be in the search path for TeX.

The indentation of the *Maxima* code can be reset by resetting the LaTeX length `\maximaindent`. The colors of the prompts, inputs and outputs can be reset by renewing the commands `\maximapromptcolor`, `\maximainputcolor` and `\maximaoutputcolor` to appropriate colors.

The top, middle and bottom of a maxima cell are determined by the commands `\maximatop`, `\maximamiddle` and `\maximabottom`. By default, `\maximamiddle` is set to `\maximaoutputmarker`. If the `lines` option is used, then `\maximatop` and `\maximabottom` are set to `\maximaboxtop` and `\maximaboxbottom`, respectively. Otherwise, `\maximatop` and `\maximabottom` are set to do nothing. The top and bottom of a maxima session are determined by `\maximasessiontop` and `\maximasessionbottom`, respectively. By default, they do nothing.

The verbatim output for maxima cells is inserted with `\maximaverbatiminput`. This will do one of two things. If the listings option is used, this is defined by `\lstinputlisting[style=emaxima]{#1}`, and so can be adjusted by resetting the emaxima style. By default, this style is given by

```
\lstdefinestyle{emaxima}
               {language=maxima,
                aboveskip=0pt,
                belowskip=0pt,
                xleftmargin=\maximaindent}
```

If the listings package is not used, then `\maximaverbatiminput` is defined by `\VerbatimInput[xleftmargin=\maximaindent]{#1}` (`\VerbatimInput` is from the fancyvrb package.) In either case, it can be adjusted by redefining `\maximaverbatiminput`, which is expected to indent everything by `\maximaindent`. The fonts used in the maximasessions is `\maximafont`, by default `\ttfamily`.