Interactive TEX/Mathematica documents

26 April 1991
For version 1.3

Dan Dill

GNU GENERAL PUBLIC LICENSE

Version 1, February 1989

Preamble

The license agreements of most software companies try to keep users at the mercy of those companies. By contrast, our General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. The General Public License applies to the Free Software Foundation's software and to any other program whose authors commit to using it. You can use it for your programs, too.

When we speak of free software, we are referring to freedom, not price. Specifically, the General Public License is designed to make sure that you have the freedom to give away or sell copies of free software, that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of a such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must tell them their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

1. This License Agreement applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any work containing the Program or a portion of it, either verbatim or with modifications. Each licensee is addressed as "you".
2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this General Public License and to the absence of any warranty; and give any other recipients of the Program a copy of this General Public License along with the Program. You may charge a fee for the physical act of transferring a copy.
3. You may modify your copy or copies of the Program or any portion of it, and copy and distribute such modifications under the terms of Paragraph 1 above, provided that you also do the following:
   - cause the modified files to carry prominent notices stating that you changed the files and the date of any change; and
   - cause the whole of any work that you distribute or publish, that in whole or in part contains the Program or any part thereof, either with or without modifications, to be licensed at no charge to all third parties under the terms of this General Public License (except that you may choose to grant warranty protection to some or all third parties, at your option).
   - If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the simplest and most usual way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this General Public License.
   - You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

   Mere aggregation of another independent work with the Program (or its derivative) on a volume of a storage or distribution medium does not bring the other work under the scope of these terms.
4. You may copy and distribute the Program (or a portion or derivative of it, under Paragraph

2) in object code or executable form under the terms of Paragraphs 1 and 2 above provided that you also do one of the following:

- accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Paragraphs 1 and 2 above; or,
- accompany it with a written offer, valid for at least three years, to give any third party free (except for a nominal charge for the cost of distribution) a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Paragraphs 1 and 2 above; or,
- accompany it with the information you received as to where the corresponding source code may be obtained. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form alone.)

Source code for a work means the preferred form of the work for making modifications to it. For an executable file, complete source code means all the source code for all modules it contains; but, as a special exception, it need not include source code for modules which are standard libraries that accompany the operating system on which the executable file runs, or for standard header files or definitions files that accompany that operating system.

5. You may not copy, modify, sublicense, distribute or transfer the Program except as expressly provided under this General Public License. Any attempt otherwise to copy, modify, sublicense, distribute or transfer the Program is void, and will automatically terminate your rights to use the Program under this License. However, parties who have received copies, or rights to use copies, from you under this General Public License will not have their licenses terminated so long as such parties remain in full compliance.
6. By copying, distributing or modifying the Program (or any work based on the Program) you indicate your acceptance of this license to do so, and all its terms and conditions.
7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein.
8. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of the license which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the license, you may choose any version ever published by the Free Software Foundation.
9. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we

sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

10. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
11. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ABSTRACT

TEX/*Mathematica* is a set of tools that provide facilities of *Mathematica* Notebooks in a UNIX environment, under GNU Emacs. They permit interaction between a text and a *Mathematica* buffer and, if desired, the use of TEX/LaTEX to annotate *Mathematica*-based explorations and programs. Inclusion of *Mathematica*-generated graphics in TEX/LaTEX documents printed using PostScript is supported. The tools also support the automatic generation of *Mathematica* packages from *Mathematica* documents.

With these tools one can interactively develop and refine teaching and research documents. The interactive nature of the tools encourages *Mathematica*-based exploration as a natural part of the writing process.

1 Introduction

Mathematica Notebooks, available on several platforms, constitute an exciting step toward the goal of interactive mathematical documents, and are a significant factor in the ready acceptance of *Mathematica* by beginning users. Unfortunately, Notebooks are not yet available on UNIX systems. This limitation is an obstacle to new users, especially students, who may be faced with the need to learn a new operating system and an editor, in addition to *Mathematica*.

Notebooks themselves have shortcomings: although they allow annotation of *Mathematica* sessions, these annotations do not support full mathematical notation in a system-independent way, as TEX, for example, does. This restricts the effective use of Notebooks in the development of course materials in upper-level mathematics courses and in courses at all levels in astronomy, chemistry, physics and in all fields of engineering, for example, since mathematics is central to their exposition.

To overcome the limitations above, I've developed tools that provide some of the facilities of Notebooks in a UNIX environment and permit the full use of TEX in *Mathematica* documents (or vice versa). The tools also support automatic generation of *Mathematica* packages from *Mathematica* documents. The primary tool is the Emacs package `tex-mathematica`, which defines a new mode for interactive exchange between a TEX document and *Mathematica*. This builds on, and extends in several ways, David Jacobson's *Mathematica* mode for Emacs (see Section 10.3 [math], page 35). In addition, macro packages are provided to format *Mathematica*-based documents with TEX.

TEX/*Mathematica* and *Mathematica* modes are in some sense complementary. Under the first, the emphasis is on interactive document preparation, so editing can be done in the text buffer. Under the second, the emphasis is on the *Mathematica* session itself, so one interacts directly with *Mathematica* in the *Mathematica* buffer. The user can switch back and forth between the two approaches as appropriate.

2 Using TEX/*Mathematica*

2.1 Starting TEX/*Mathematica*

To start up TEX/*Mathematica*, use the command `M-x tex-mathematica`. This will set up `TeX-Mathematica mode` in the working buffer. *Mathematica* will be started as a shell process in `Mathematica mode` in a second buffer (but not until necessary). The command `C-c o` switches to the *Mathematica* buffer in another window. The command `C-c l` places the *Mathematica* input prompt at the top of the *Mathematica* window.

The command `M-x tex-mathematica` tries to determine whether the buffer contains TEX or LaTEX text, and then sets the mode accordingly to `TeX/Mathematica mode` or `LaTeX/Mathematica mode`. For cases where this results in the wrong mode, the commands `M-x plain-tex-mathematica` and `M-x latex-mathematica` are provided to explicitly select the mode.

Note that `TeX/Mathematica mode` and `Mathematica mode` are complimentary approaches to interacting with *Mathematica*. In `TeX/Mathematica mode` the emphasis is on interactive document preparation and so editing can be done in the TEX buffer. In `Mathematica mode` the emphasis is on the *Mathematica* session itself and so one interacts directly with *Mathematica* in the *Mathematica* buffer. Because `TeX/Mathematica mode` uses `Mathematica mode`, both approaches can be used as appropriate, simply by changing buffers.

2.2 *Mathematica* cells

At the simplest level, all text delineated by blank lines can be sent to the *Mathematica* buffer with the command `C-c RET`. This command is convenient for quickly sketching ideas. However, the bulk of TEX/*Mathematica* depends on its *cell structure*.

2.2.1 Regular cells

Mathematica code in a TEX document is contained in *cells*, delineated by

```
\begin{mathematica}
...
```

```
\end{mathematica}
```

The command `C-c c` creates an active cell and the commands `C-c C-[` and `C-c C-]` move to preceding and subsequent cells. Cell input is sent to *Mathematica* with the command `ESC RET` and the output of the last *Mathematica* command is inserted into the cell with the command `C-c r` (replace). These two operations, send and replace, are combined with the command `C-c u` (update).

For example, here is a *Mathematica* cell containing just input:

```
\begin{mathematica}
Apply[Plus, Array[{{#,#^3},{#^2,1}}&, 100]] // MatrixForm
\end{mathematica}
```

When this cell is formatted by TEX it is delineated by horizontal rules:

```
Apply[Plus, Array[{{#,#^3},{#^2,1}}&, 100]] // MatrixForm
```

Here is the cell after `C-c u` (update):

```
\begin{mathematica}
Apply[Plus, Array[{{#,#^3},{#^2,1}}&, 100]] // MatrixForm

                       5050       25502500

Out[1]//MatrixForm= 338350     100
\end{mathematica}
```

The output is separated from the input by a blank line, and contains the output marker `Out[...`, if any. Here is how the cell appears formatted by TEX:

```
Apply[Plus, Array[{{#,#^3},{#^2,1}}&, 100]] // MatrixForm

                    5050      25502500

Out[1]//MatrixForm= 338350    100
```

The commands `C-c a` and `C-u C-c a` evaluate *all* cells in a document (except cells that refer to other cells, see Chapter 3 [Packages], page 11; these can be evaluated using `C-u C-c m`, see Section 3.2 [Assembling], page 13). The first form lets a cell be bypassed optionally.

See Section 2.2.2 [Initialization cells], page 9 for how to mark only certain cells for evaluation.

See Section 4.2 [Customization], page 16 for information on changing the appearance of formatted cells.

2.2.2 Initialization cells

An important feature of *Mathematica* Notebook frontends is the ability to specify that certain cells should be evaluated as initialization, typically when a document is first opened. Such *initialization cells* are specified in TEX/*Mathematica* documents as

```
\begin{mathematica}[* Initialization Cell *]
...
\end{mathematica}
```

The command `C-c TAB` toggles the initialization indicator. The commands `C-c i` and `C-u C-c i` evaluate all initialization cells in a document (except cells that refer to other cells, see Chapter 3 [Packages], page 11; thses can be evaluated using `C-u C-c m`, see Section 3.2 [Assembling], page 13). The first form lets an initialization cell be bypassed optionally. Here is an example of an initialization cell:

```
\begin{mathematica}[* Initialization Cell *]
u[t_] := {{Cos[t], -Sin[t]}, {Sin[t], Cos[t]}}; \
v = {{0,I}, {-I,0}};
\end{mathematica}
```

Note that these particular commands generate no *Mathematica* output and so there is no output portion to the cell. In the formatted document an initialization cell is marked with a boxed `I` at the right edge of the cell-start rule:

$\boxed{\texttt{I}}$

```
u[t_] := {{Cos[t], -Sin[t]}, {Sin[t], Cos[t]}}; \
v = {{0,I}, {-I,0}};
```

2.3 Arrangement of *Mathematica* statements in cells

Statements may be set off with `TAB` characters, and these will be reflected in the formatted document, based on an eight-character tab width.

Cell input can contain more than one *Mathematica* statement, and each statement can extend across several lines. Multiple statements must be separated by '`;`' and when *Mathematica* statements extend over multiple lines, any lines except the last that could be interpreted by *Mathematica* as a valid end of statement must end with the continuation character '`\`'. The '`;`' and '`\`' are necessary because each line of input is sent separately to *Mathematica*. For example, consider the cell

```
y = u[Pi] . v \
       /. 0 -> x

Out[3]= {{x, -I}, {I, x}}
```

Without the '`\`', *Mathematica* would process the first line as a complete statement and then report an error for the next line. In fact `tex-mathematica` will flag such potential problems (see Section 6.2 [Mathematica help], page 24).

3 Creating *Mathematica* packages

You may want to collect some of the *Mathematica* cells of a TeX/*Mathematica* document into a *Mathematica* package. This chapter describes how to do this automatically.

This feature of TeX/*Mathematica* is modeled on Knuth's WEB system of structured documentation. The basic idea of the WEB system is to develop code hierarchically in terms of named fragments. Fragments are introduced as appropriate, in whatever order is convenient to the description of the program, using TeX for documentation. Thereby the author is freed from the constraints of the programming language in the development and description of the code.

In the usual implementation of WEB systems, the source document is written in WEB, the mixture of code and TeX documentation. The WEB document is then alternatively processed by a program WEAVE into a file which can then be processed by TeX to give the formatted documentation and code, or by a program TANGLE which collects together the code fragments into a source file for the particular language processor (for example, Pascal or C.

Here instead, since *Mathematica* code is printed verbatim without other special formatting, the source TeX/*Mathematica* document can be processed directly by TeX (or LaTeX), that is, the WEAVE step is not needed. The generation of the *Mathematica* package code, the TANGLE step, is done either interactively from within Emacs or from a UNIX shell.

3.1 Marking *Mathematica* cells as part of a package

A cell is marked as part of a *Mathematica* package by appending a *cell label* <*file*:*part*> to the end of the line starting the *Mathematica* cell:

```
\begin{mathematica}<file:part>
...
\end{mathematica}
```

where *file* is the name of the file to contain the text of the *Mathematica* package and *part* specifies to what part of the package the cells belongs. The order in which parts are assembled into the package file (see Section 3.2 [Assembling], page 13) is determined by package references contained in *package cells*, cells marked <*file*:> — without a part. Cells referenced in package cells can also contain cell references and the same cell reference can be used in multiple cells. In this way the structure of the package can be set out hierarchically.

Here is an example of how this works. Let's say we want to assemble the package file '`MatrixMechanics.m`'. The overall structure of the package might be defined by the package cell

```
\begin{mathematica}<MatrixMechanics.m:>
BeginPackage["MatrixMechanics.m`"]

<MatrixMechanics.m:Usage messages>

Begin["`Private"]

<MatrixMechanics.m:Error messages>

<MatrixMechanics.m:Utility functions>

<MatrixMechanics.m:Function QMmatrix>

End[]; EndPackage[]
\end{mathematica}
```

in terms of parts `Usage messages`, `Error messages`, `Utility functions` and `Function QMmatrix`. These parts are defined in correspondingly marked cells. For example, part `Function QMmatrix` might have the definition

```
\begin{mathematica}<MatrixMechanics.m:Function QMmatrix>
QMmatrix[op_, basis_List] := Block[{
        <MatrixMechanics.m:QMmatrix local variables>
        },
    <MatrixMechanics.m:QMmatrix argument checking>
    <MatrixMechanics.m:QMmatrix evaluation>
\end{mathematica}
```

in terms of still other parts. Cells referenced in package cells can also contain cell references and the same cell reference can be used in multiple cells. This is convenient to control the layout of the assembled package.

Cells with the same label will be collated together in the order in which they appear in the document (see Section 3.2 [Assembling], page 13). For example, the part `Utility functions` could be built up from several cells

```
\begin{mathematica}<MatrixMechanics.m:Utility functions>
...
\end{mathematica}
```

When marked cells are formatted by TEX, the package filename and partname are printed in the cell start rule, separated by a colon, for example,

```
<MatrixMechanics.m:Function QMmatrix>
...
```

Package cells are marked `Definition of package` *file*, for example,

```
Definition of package MatrixMechanics.m
...
```

It is useful to make package filenames and, especially, partnames as descriptive as possible. This is made easier by name completion and name abbreviation. The command `ESC TAB` will complete filenames and partnames based on names appearing elsewhere in the document (see Section 6.2 [Mathematica help], page 24 for another use of `ESC TAB`). The filename and partname also can be abbreviated by a unique substring followed by `...`, for example

```
\begin{mathematica}<Matrix...:U...>
...
\end{mathematica}
```

If only a single package filename or partname appears in the document, then it may be abbreviated to simply `...`. Thus, utility functions of the package file '`MatrixMechanics.m`' could be specifed in the same TEX/*Mathematica* document as (assuming only the single package filename '`MatrixMechanics.m`' appears in the document for cell-type `mathematica`; documents may contain other cell-types, see Chapter 8 [Extending tex-math], page 27):

```
\begin{mathematica}<...:U...>
...
\end{mathematica}
```

The abbreviation marker `...` appears in the cell start rule and so can leave space there for long partnames.

3.2 Assembling a package

The command `C-c m` (`M-x tex-mma-assemble-package`) assembles cells marked as part of a package. The package is assembled in a buffer of the same name as the specified filename. If only a single package filename is specified in the TEX/*Mathematica* document, then it will be used. Otherwise the command will prompt for the package filename.

The assembly of the package proceeds as follows: First, all cells for each part are collated together in the order they appear in the document; package cells are also collated together. Then the combined text for each part is collated together in the order the references to it appear in the collated package cell text (see Section 3.1 [Marking], page 11). Finally, the text for any resulting references is inserted, recursively until all references have been resolved.

You can also assemble any single marked cell, using the command `C-u C-c m` (`M-x tex-mma-assemble-cell`). The text is assembled in the buffer *file:part* corresponding to the cell label. The assembly proceeds as for package cells, except that only cells referred to in the particular cell are collated; other cells with the same cell label are *not* collated. Assembly of an individual cell is useful for checking intermediate phases of package assembly and for preparing a cell for submission to *Mathematica*.

Note the difference between assembling a cell and assembling a package. If you specify just a single package cell in your document, then you can assemble the package cell with either `C-c m` or `C-u C-c m`. If your package is specified in several package cells, then you need to use `C-c m` so that their text will be collated together in the package assembly.

3.3 Assembling a package from the UNIX shell

You can create a package file from the UNIX shell, for example in conjunction with UNIX make to ensure that packages generated from a TEX/*Mathematica* document reflect any changes to the document. You can do this with the UNIX command `tex-mma-assemble-package` (a shell script), which runs the Emacs Lisp function `tex-mma-assemble-package` using Emacs batch mode. The UNIX command takes three arguments:

`tex-mma-assemble-package` *package-type package-file document-file*

the package type (usually `mathematica`, but other kinds of cells and so packages are also possible, see Chapter 8 [Extending tex-math], page 27), the package filename, and the TEX/*Mathematica* document filename.

For example, let's say you want to create the package file `AngularMomentum.m` from marked cells in the LaTEX document '`am.tex`'. You can do this with the UNIX command

```
tex-mma-assemble-package mathematica AngularMomentum.m am.tex
```

This command tells Emacs to visit '`am.tex`', assemble the package file '`AngularMomentum.m`' from marked *Mathematica* cells and then exit. Note that '`AngularMomentum.m`' will be overwritten if necessary. The command also creates the file '`AngularMomentum.m.log`' that contains what would have been printed in the echo area had the package been assembled interactively from within Emacs (using `C-c m`).

4 Formatting TEX/*Mathematica* documents

One reason for creating this package was so you can easily edit, annotate and save work with *Mathematica*. You can do this without having to use TEX at all, just by using the cell creation, send, update and replace commands (see Section 2.2.1 [Regular cells], page 7), together with the help and syntax error facilities (see Section 6.2 [Mathematica help], page 24). If you want to use TEX to format your document, however, macro packages are provided for formatting *Mathematica* cells.

4.1 Initialization

In 10 point TEX documents *Mathematica* cell formatting is made available with

```
\input mathematica10pt
```

Use `mathematica12pt` for 12 point TEX documents. In LaTEX use

```
\documentstyle[mathematica,...
```

for both 10 and 12 point documents. With these macro packages *Mathematica* sessions can be neatly integrated into TEX documents.

4.2 Customizing cells

The type style, type size (for LaTEX only), and the indentation of cell text can be changed.

4.2.1 Type style

Cells are set in a fixed-width typewriter font. This is necessary to preserve tabular alignment. Any fixed-width font can be used. To change the font, the macro files need to be changed. For TEX, in file '`mathematica10pt.tex`', change the `cmtt8<` in the line

```
\font\cellf@nt=cmtt8 % Font for cell text
```

or in file '`mathematica12pt.tex`' change the `cmtt9` in the line

```
\font\cellf@nt=cmtt9 % Font for cell text
```

Make sure that the new font style is the same size (that is, 8 or 9 points) or the tabbing and line spacing will come out wrong. For LaTeX, in '`mathematica.sty`' change the `\tt` in the line

```
\def\v@rbsize{\mm@size\tt} % Font for cell text
```

4.2.2 Type size

For LaTeX documents (but not TeX documents) the type size can be changed with the command

```
\mmasize{<type size specification>}
```

The default size is `\footnotesize`. Use `\normalsize` to make cell text the same size as regular text. In 12 point documents with narrow margins `\scriptsize` might work well,

```
\mmasize{\scriptsize}
```

4.2.3 Indentation

For both LaTeX and TeX, the left margin of cell contents can be changed with the command

```
\mmaindent{<size specification>}
```

The default indentation is `\parindent`. For example, to set the cell margin flush left, use

```
\mmaindent{0pt}
```

4.3 Example TeX/*Mathematica* text

The following paragraph, taken from lecture notes on quantum mechanics, illustrates how TeX and *Mathematica* can be combined.

Clebsch-Gordan coefficients $\langle j_1 m_1, j_2 m_2 | jm\rangle$ arise in the quantum theory of angular momentum. They are elements of unitary transformation between uncoupled and coupled representations $|j_1 m_1 j_2 m_2\rangle$ and $|j_1 j_2 jm\rangle$. The unitarity of the transformation implies the sum rules

$$\sum_{jm} \langle j_1 m_1, j_2 m_2 | jm\rangle \langle jm | j_1 m_1, j_2 m_2\rangle = 1,$$
$$\sum_{jm} \langle jm | j_1 m_1, j_2 m_2\rangle \langle j_1 m_1, j_2 m_2 | jm\rangle = 1.$$

The first of these rules can be illustrated explicitly as follows. First,

```
<<DiscreteMath/ClebschGordan.m
```

defines the function `Clebsch[j1,m1,j2,m2,j,m]`.Next, to keep things simple, consider the case $j_1 = j_2 = l$, so that $0 \le j \le 2l$, and $m_1 = m_2 = 0$, so that $m = 0$. Then the only coefficients needed are

```
cg[l_, j_] := Clebsch[{l,0},{l,0},{j,0}]
```

and the first sum rule can be written

```
check[l_] := 1 == Sum[cg[l,j]^2, {j, 0, 2 l}]
```

The rule for $l = 0$ to $l = 5$ is then checked with

```
Table[check[i], {i, 0, 5}]

Out[7]= {True, True, True, True, True, True}
```

The second sum rule can be tested in a similar way.

Using TEX/*Mathematica* in this way, one can interactively develop and refine course and research documents. The interactive nature of these tools encourages exploration as a natural part of the writing process. At every point a fully annotated record of the journey so far is at hand.

4.4 Verbatim listing of *Mathematica* (and other) files

You can insert the verbatim listing of *Mathematica* file <filename> (or any text file, for that matter) in your document with the command

```
\mathematicafile{<filename>}
```

The file contents will have the same format as cells and the file name will be included in the cell-start rule.

Caution: If text follows on the next line, then include the file with the command

```
\mathematicafile{<filename>}%
```

The trailing % prevents a space before the start of the text on the next line. If the next line is empty, then the trailing % is not needed (but will do no harm).

As an example, the command

```
\mathematicafile{PSTeX.m}
```

inserts the verbatim listing of the file 'PSTeX.m', which defines the command PSTeX (see Section 5.2 [Files], page 21):

File PSTeX.m

```
(* 2/25/1991 Dan Dill dan@chem.bu.edu *)

(* This version assumes the Mathematica prolog file will be avilable in the
   printer.  The current prolog is mma.pro.1.2 and is specific to version 1.2
   of Mathematica.  It will change in version 2.0.  The fontsize (at
   PSTeXHeight) is set as PSTeXFontsize  to 7 bp (72 bp per inch).  PSTeXpro.m
   is a version of PSTeX that includes the prolog in the file, using psfix,
   and so should work unchanged in version 2.0. *)

PSTeX::usage = "PSTeX[-graphics-, \"file\"].  Process a graphics object to a
PostScript file for inclusion in a TeX document (using \\psfig).  Quotes are
required around the file name and suffix `.ps' is added.  The graphic object
is scaled and shifted so that it can be included in the TeX document directly
(for example, with \\psfig{figure=file.ps}).  The height and font size of the
scaled graphic are PSTeXHeight (default 100) and PSTeXFontsize in big points
(72 bp per inch). For example,
\n
\n
\t PSTeX[myplot, \"p4/myplot\"]
\n
\n
  will convert the graphics object myplot to the PostScript file myplot.ps in
the subdirectory p4 of the current directory.
```

```
\n
\n
  Notes: The Mathematica PostScript prolog is *not* included in the file and
the prolog must be downloaded before the file can be printed.  PSTeX yields a
graphic with PSTeXHeight and without centering or margins, and so is not
generally suitable for printing.  Use PSFile to produce a graphic in a file,
centered and scaled to fill the whole page."

PSTeX::file = "Graphics being processed (without prolog) to file \"`1`\".";
PSTeXHeight = 100;
PSTeXFontsize = 7;
PSTeX[g_, f_String] := Block[{filename},
       filename = StringJoin[f, ".ps"];
       Message[PSTeX::file, filename];
       Display[StringJoin[
              "! ",
              "addBBox",
              " ",
              ToString[PSTeXHeight],
              " ",
              ToString[PSTeXFontsize],
              " > ",
              filename
              ], g];];
```

5 *Mathematica* graphics

Graphics generated by *Mathematica* can be incorporated into TEX documents using Trevor Darrell's `psfig` TEX/LaTEX macros. These are described fully in the detailed documentation provided as part of the `psfig` kit (see Section 10.4 [psfig], page 35). (The `epsf` macros of Tomas Rokicki's program `dvips` can be used instead; See Section 10.2 [dvips], page 34.) Including graphics in a TEX/*Mathematica* document is a three-part process. First the `psfig` macro must be read in, then graphics are generated to files with appropriate PostScript bounding box information, and finally files are placed in the document.

5.1 Preparing a document for graphics

Before *Mathematica* graphics files can be included in a document, the `psfig` macro (see Section 10.4 [psfig], page 35) must be read in and initialized. For a TEX document use

```
\input psfig
...
\psfiginit
```

For a LaTEX document use

```
\documentstyle[mathematica,psfig,...
...
\begindocument
...
\psfiginit
```

The command `\psfiginit` must come before any use of `\psfig`; in LaTEX documents `\psfiginit` also must come *after* the `\begin{document}`.

5.2 Generating *Mathematica* graphics to a file

Figures are incorporated via PostScript graphics files that contain the proper bounding box information. The *Mathematica* command `PSTeX[-graphics-, "file"]`, defined in '`PSTeX.m`' (see Section 9.2 [Tool kit], page 30), generates these graphics files.

Note that the *Mathematica* PostScript prolog is *not* included in the file and the prolog must be downloaded to the printer before the file can be printed. This is done by the `dvips` processor (see Section 10.2 [dvips], page 34), using `\special` commands inserted in the '`.dvi`' file by `\psfiginit` (see Section 5.1 [Setup], page 21).

Double quotes are required around the file name and suffix '`.ps`' is added. Use `ESC ‘ "` (`M-x quoted-insert "`) to insert a double quote in a TEX buffer. The graphic is scaled to height 100 points (72 points equal 1 inch), using a scheme developed by Cameron Smith. The height can be changed by adjusting the value of `PSTeXHeight` or by using the `height` variable of `\psfig`.

`PSTeX` yields a graphic scaled to height `PSTeXHeight` and without centering or margins and hence not generally suitable for printing. `PSFile`, defined in '`PSFile.m`' (see Section 9.2 [Tool kit], page 30), produces a graphic in a file, centered and scaled to fill the whole page. The font size is 7 points at the height `PSTeXHeight`; this can be changed by changing `PSTeXFontsize`. Note that the `height` variable of `psfig` results in the whole graphic being scaled and so the font size as well.

If `PSTeX` is not yet set up (say by the *Mathematica* initialization file '`init.m`'), then it is defined with

```
<<PSTeX.m
```

Once `PSTeX` has been set up, a figure is generated to a file with commands like

```
fig = Plot[Sin[x]^3, {x, 0, 4Pi}]; \
        PSTeXFontsize = 9; \
        PSTeX[fig, "sin3x"];
```

Note that in this example the the font size has been changed from the default (7 points) to 9 points.

5.3 Placing graphics in documents

5.3.1 LaTEX documents

Figures can be placed in a LaTEX document using the `\figure` environment. For example

```
\begin{figure}
\centerline{\psfig{figure=sin3x.ps}}
\caption{{\sl Mathematica} plot of $\sin^3 x$.}
\end{figure}
```

5.3.2 TEX documents

Figures can be placed in a TEX document using the various *insert* commands. For example, here is a macro using `\midinsert` to place figures and give them a caption.

```
% #1 is the file
% #2 is figure height in the printed document
% #3 is the figure caption
\def\insertfig#1#2#3{
    \midinsert\centerline{%
        \psfig{figure=#1,height=#2}}
        \global\advance\figno by1 % \global so \figno incremented
        \setbox0=\vbox{%
            \noindent
            \leftskip=.2\hsize
            \rightskip=.2\hsize
            {\bf Figure \the\figno}. #3}
        \vskip 18pt
        \centerline{\box0\hss}
        \vskip 24pt
    \endinsert
}

% Define and initialize figure number
\newcount\figno \figno=0
```

Placing the caption (macro argument 3) in a `\vbox` allows the caption to extend over many lines. Here is an example of how to use this macro:

```
\insertfig{sin3x.ps}{100pt}
{{\it Mathematica} plot of $\sin^3x$.}
```

6 Getting help

On-line help is available for both `Emacs` and *Mathematica*.

6.1 Help on `tex-mathematica` and `math`

This Info description of the facilities of TEX/*Mathematica* is accessed by the command `C-c h`. The version of TEX/*Mathematica* that is running is displayed with the command `C-c v`. Help on the commands available in the TEX/*Mathematica* and *Mathematica* buffers is given by the command `C-h m` in each buffer.

6.2 Help on *Mathematica*

David Jacobson's GNU Emacs package `math` contains several aids that have been made available in the TEX/*Mathematica* buffer: automatic syntax checking of *Mathematica* input, normal and extended help on *Mathematica* symbols (`C-h e` and `C-h E`), name completion (`ESC TAB`) (see Section 3.1 [Marking], page 11 for another use of `ESC TAB`), and location of syntax errors in '`.m`' files (`C-c C-e`). In particular, the input syntax checking detects the problem in the above cell caused by omitting the '\' (see Section 2.3 [Input], page 9).

Since TEX/*Mathematica* uses `Mathematica mode`, the help, name completion and '`.m`'-file syntax error location commands also work directly from the *Mathematica* buffer.

7 Customizing TEX/*Mathematica*

This chapter documents the TEX/*Mathematica* commands defined in the TEX/*Mathematica* buffer and how their key bindings can be changed.

7.1 TEX/*Mathematica* key bindings

Here are all of the commands of TEX/*Mathematica* defined in the TEX/*Mathematica* buffer and the Emacs Lisp functions that they invoke. The bindings are defined in the key map `TeX-mma-mode-map`.

```
C-c C-[        tex-mma-backward-cell
C-c C-]        tex-mma-forward-cell
C-c C-b        TeX-buffer
C-c C-e        tex-mma-math-find-error
C-c C-f        TeX-close-LaTeX-block
C-c C-k        TeX-kill-job
C-c C-l        TeX-recenter-output-buffer
C-c C-p        TeX-print
C-c C-q        TeX-show-print-queue
C-c C-r        TeX-region
C-c RET        tex-mma-send
C-c TAB        tex-mma-toggle-init
C-c a          tex-mma-eval-all-ask
C-c c          tex-mma-create-cell
C-c h          tex-mma-info
C-c i          tex-mma-eval-init-ask
C-c l          tex-mma-recenter
C-c m          tex-mma-assemble-package
C-c r          tex-mma-replace
C-c s          tex-mma-show
C-c u          tex-mma-update
C-c v          tex-mma-version

C-h E          math-extra-help
C-h e          math-help

C-u C-c a      tex-mma-eval-all-noask
C-u C-c i      tex-mma-eval-init-noask
C-u C-c m      tex-mma-assemble-cell

ESC RET        tex-mma-send-cell
ESC TAB        tex-mma-do-completion
ESC o          tex-mma-buffer-go
```

```
ESC {           TeX-insert-braces
ESC }           up-list
```

The commands beginning `tex-mma-` are defined in '`tex-mma.el`' (see Section 9.2 [Tool kit], page 30), the commands beginning `TeX-` are defined in '`tex-mode.el`' (part of GNU Emacs) and the commands beginning `math-` are defined in '`math.el`' (see Section 9.2 [Tool kit], page 30).

7.2 Changing key bindings

If you want to change some key bindings, add additional key bindings of your own, or perform any other operations when starting up TEX/*Mathematica*, you can do so by defining the variable `TeX-mma-mode-hook` in your initialization file '`.emacs`'. Here is an example. It defines the cell motion commands to be `C-c [` and `C-c ]`.

```
(setq TeX-mma-mode-hook
      '(lambda ()
         (define-key TeX-mma-mode-map "\C-c[" 'tex-mma-backward-cell)
         (define-key TeX-mma-mode-map "\C-c]" 'tex-mma-forward-cell)
         ))
```

Commands are defined in the TEX/*Mathematica* mode key map `TeX-mma-mode-map` (see Section 7.1 [Bindings], page 25).

8 Extending TEX/*Mathematica*

By default TEX/*Mathematica* defines "mathematica" cells as its single cell type. You can define additional cell types. This chapter describes how to do this.

8.1 Emacs Lisp specification of new cell types

The file '`unix-tex-mma.el`' provided with TEX/*Mathematica* is an example Emacs Lisp implementation of cell type "unix", for interaction between the TEX buffer and a UNIX shell running in another buffer. There are three things that the implementation must contain:

1. The function `'type'-tex-mma-hook`, which is used to make "type" cells part of TEX/*Mathematica*. For example,

   ```
   (defun unix-tex-mma-hook ()
     "Hook for TeX/Mathematica."
     ; Put any setup processing you need to do here.  You don't have to
     ; do any, but this function must still be defined.
     )
   ```

   Whether or not you define a body for this function, the function itself must still be defined. It will be used by Emacs to make sure the functions for this cell type are loaded, using the `autoload` mechanism (see Section 8.2 [Init], page 28).
2. The string `'type'-tex-mma-version-string` describing the version of this cell interface. For example,

   ```
   (defconst unix-tex-mma-version-string
     "TeX/Mathematica UNIX shell interface Version 1 <Thur 11 April 1991>"
     "String describing this version of TeX/Mathematica UNIX interface.")
   ```

   This string will be displayed by `C-c v` to identify the cell types supported by the version of TEX/*Mathematica* that is running.
3. The statement (for example)

   ```
   (defconst unix-tex-mma-cell-alist
     '(
       ("buffer-go" . unix-buffer-go)
       ("recenter" . unix-recenter)
       ("replace" . unix-replace)
       ("send" . unix-send)
       ("send-cell" . unix-send-cell)
       ("show" . unix-show)
       ("update" . unix-update)
       ))
   ```

to associate the cell type `unix` with the functions that communicate with its buffer. The list entries define the functions that replace the corresponding functions of TEX/*Mathematica*. The statement above includes entries for all TEX/*Mathematica* operations that use the *Mathematica* buffer. There should be an entry for each function you want defined for your cell type. If an entry is absent, the corresponding operation will be ignored for your cell type. You can define additional functions for your cell type. These do not need to be in the list, but their presence will cause no harm.

8.2 Modifications to the Emacs init file

For each piece you want to add to TEX/*Mathematica* you should use an `autoload` function in, for example, your Emacs init file to associate its functions `‘type’-tex-mma-hook` (see Section 8.1 [Interface], page 27) with the Emacs Lisp file in which the functions for the type are defined. For example, to add cell types "c" and "unix", defined in (say) ‘`c-tex-mma.el`’ and ‘`unix-tex-mma.el`’, you need

```
(autoload 'c-tex-mma-hook "c-tex-mma"
          "Add `c' cells to TeX/Mathematica.")
(autoload 'unix-tex-mma-hook "unix-tex-mma"
          "Add `unix' cells to TeX/Mathematica.")
```

Then, you must list all of the cell types that you are adding in the list `tex-mma-new-types`. For example,

```
(defconst tex-mma-new-types '("c" "unix"))
```

tells TEX/*Mathematica* to add cell types "c" and "unix".

8.3 Modifications to TEX/LaTEX macros

You will have to add your new cell type(s) to the TEX/LaTEX macros if you want to format your document. The file ‘`mathematica.tex`’ contains code common to TEX and LaTEX for the actual formatting of cell. The file ‘`mathematica.sty`’ handles the LaTEX interface to ‘`mathematica.tex`’. The files ‘`mathematica10pt.tex`’ and ‘`mathematica12pt.tex`’ provide the TEX interface. You will have to adapt the macros in these file to handle your cells.

At a minimum you will have to define your `\begin{‘type’} ... \end{‘type’}` commands.

Then you can format your new cells in the same way as *Mathematica* cells. You might also want to define new cell rules to identify your cells, using the *Mathematica* cell rules as a starting point.

9 Getting TEX/*Mathematica*

The TEX/*Mathematica* tools are available from Internet host `chem.bu.edu [128.197.30.18]` by anonymous `ftp` in directory `/pub/tex-mathematica`. The author can be reached at Internet address `dan@chem.bu.edu`.

The `ftp` directory contains four files

- '`README`', which duplicates the information in this chapter.
- '`CHANGES`', which describes changes that have been made.
- '`tex-mma-j.ps.Z`', the compressed PostScript documentation/example.
- '`tex-mma.tar.Z`', the TeX/Mathematica distribution kit (includes the first three files).

9.1 LaTEX description of TEX/*Mathematica*

The PostScript document '`tex-mma-j.ps.Z`' is derived from '`tex-mma-j.tex`'. Transfer the document from the `ftp` directory in binary (image) mode and then print it with

```
zcat tex-mma-j.ps.Z | lpr
```

The source for this document is the LaTEX file '`tex-mma-j.tex`' and the BibTEX file '`tex-mma-j.bib`'. You can use '`tex-mma-j.tex`' as a LaTEX example of a TEX/*Mathematica* document.

9.2 Files in the distribution kit

The file '`tex-mma.tar.Z`' contains the TEX/*Mathematica* tools. These consist of the following:

Documentation things:

- '`tex-mma.texinfo`', Texinfo documentation of TEX/*Mathematica*
- '`tex-mma.ps.Z`', compressed Texinfo document (this document) formatted with TEX into PostScript
- '`tex-mma.info`', Info file for on-line `Emacs` documentation
- '`tex-mma-j.tex`', LaTeX description/example of TEX/*Mathematica*

- '`tex-mma-j.bib`', BibTEX file for '`tex-mma-j.tex`'
- '`tex-mma-j.ps.Z`', compressed LaTEX documentation formatted in PostScript
- '`sin3x.ps`', *Mathematica*-generated figure included in '`tex-mma-j.tex`'
- '`tex-mma-tex.tex`', TEX example of TEX/*Mathematica*
- '`texinfo.tex`', TeX macros used to format '`tex-mma.texinfo`'

GNU Emacs things:

- '`tex-mma.el`', the `tex-mathematica` package
- '`math.el`', David Jacobson's *Mathematica* mode package (see Section 10.3 [math], page 35)
- '`unix-tex-mma.el`', example implementation of cell-type `unix`

TEX/LaTEX things:

- '`mathematica10pt.tex`', TEX 10 point TEX/*Mathematica* interface
- '`mathematica12pt.tex`', TEX 12 point TEX/*Mathematica* interface
- '`mathematica.sty`', LaTEX generic TEX/*Mathematica* interface
- '`mathematica.tex`', macros for formatting TEX/*Mathematica* documents, used by the preceding files

Shell scripts and *Mathematica* commands for processing graphics, and shell script and template file for *Mathematica* package assembly:

- '`PSTeX.m`', generates `psfig`-adapted graphics to a file, without PostScript prolog
- '`addBBox`', shell script called by `PSTeX`
- '`PSTeXpro.m`', generates `psfig`-adapted graphics to a file, with PostScript prolog
- '`addBBoxpro`', shell script called by `PSTeXpro`
- '`addBBoxpro.awk`', `awk` script called by '`addBBoxpro`'
- '`PSFile.m`', generates full-page graphics to a file, with PostScript prolog
- '`tex-mma-assemble-package`', shell script for *Mathematica* package assembly
- '`tex-mma-assemble-package.tmplt`', Emacs Lisp template for '`tex-mma-assemble-package`'

This is from Cameron Smith (see Section 10.1 [addBBox], page 34):

- '`mma.pro.1.2`', *Mathematica* Version 1.2 PostScript prolog. Note that a different prolog will be needed for *Mathematica* Version 2.0. This will be available from the author (see Chapter 9 [Getting tex-math], page 30).

These are from Trevor Darrell's `psfig/tex` distribution (see Section 10.4 [psfig], page 35):

- '`psfig.pro`', `psfig` PostScript prolog.
- '`psfig.sty`', `psfig` for LaTEX.
- '`psfig.tex`', `psfig` for TEX.

9.3 Installation procedure

1. Transfer '`tex-mma.tar.Z`' in binary (image) mode into an empty directory, and extract its contents, with (for example)

   ```
   zcat tex-mma.tar.Z | tar xvf -
   ```

2. Install the Info on-line documentation file '`tex-mma.info`' where your Emacs looks for Info file. If desired, add a pointer to it to your Info directory file so that it will appear in the Info top-level menu. The source for '`tex-mma.info`' is '`tex-mma.texinfo`'. The file '`tex-mma.ps.Z`' is the compressed TEX formatted version, made using the '`texinfo.tex`' included in the distribution (see the beginning of '`tex-mma.texinfo`' for directions). The formatted version can be printed with

   ```
   zcat tex-mma.ps.Z | lpr
   ```

   It contains complete details of the TEX/*Mathematica* tools.

3. Edit `tex-mma.el` to
   - set `tex-mma-process-string` to the command you use to start *Mathematica* (default is `math`);
   - set `tex-mma-info-file` to point to where you put the file '`tex-mma.info`'
4. Run `M-x byte-compile` on '`tex-mma.el`' and store the resulting '`tex-mma.elc`' wherever your GNU Emacs looks for files.
5. Add the following to the GNU Emacs initialization file

   ```
   (autoload 'tex-mathematica "tex-mma"
             "Major-mode for interaction with Mathematica from TeX." t)
   (autoload 'plain-tex-mathematica "tex-mma"
             "Major-mode for interaction with Mathematica from TeX." t)
   (autoload 'latex-mathematica "tex-mma"
             "Major-mode for interaction with Mathematica from TeX." t)
   ```

6. If you do not have David Jacobson's *Mathematica* mode package '`math.el`', install it following the instructions in '`math.el`'. If you do have '`math.el`', make sure your version is at least as current as the one here, which contains changes necessary for use with '`tex-mma.el`'. Make sure you place the file '`math.el`' (or, better, the byte-compiled file '`math.elc`') where your Emacs looks for libraries, so that '`tex-mma.elc`' will be able to load it if it is not already loaded.

7. Edit '`mathematica.tex`' (at the end) to specify where the PostScript prolog files '`psfig.pro`' and '`mma.pro.1.2`' will be, and put the prolog files there.
8. If you are using a LaTEX *earlier* than 2 May 90, edit '`mathematica.sty`' according to the comments there.
9. Put '`mathematica.tex`', '`mathematica.sty`', '`mathematica12pt.tex`', '`mathematica10pt.-tex`', '`psfig.sty`', and '`psfig.tex`' where your TEX looks for macro files.
10. Put '`PSTeX.m`', '`PSTeXpro.m`' and '`PSFile.m`' where your *Mathematica* looks for packages.
11. Edit '`addBBoxpro`' to specify which `awk` you use (default is GNU awk, `gawk`).
12. Edit '`tex-mma-assemble-package`' to set the variables `bindir`, `tmpdir` and `tmpdirsed` for your system, as described in the comments there. Then place '`tex-mma-assemble-package.tmplt`' in `bindir`.
13. Put '`addBBox`', '`addBBoxpro`', '`addBBoxpro.awk`' and '`tex-mma-assemble-package`' on your system's binary search path, make sure they have execute status, and then execute `rehash`.

You should be ready to go: Startup emacs, run `M-x tex-mathematica` (see Section 2.1 [Startup], page 7), and have fun.

10 Related programs

TEX/*Mathematica* uses several other programs. All except `dvips` are included in the TEX/*Mathematica* distribution, and `dvips` is freely available. These programs and where they come from are summarized here.

10.1 `addBBox`: PostScript bounding box shell script

`addBBox` is the UNIX shell script that is called by `PSTeX` to adjust the PostScript bounding box in *Mathematica* graphics so that the resulting graphics files can be handled properly by `psfig`. `addBBox` is based on a scheme developed by Cameron Smith. His electronic mail address is `cameron@midd.cc.middlebury.edu`.

10.2 `dvips`: TEX-dvi-to-PostScript processor

`dvips` is a program that converts TEX '`.dvi`' (device-independent) files into PostScript. Several such programs are available. TEX/*Mathematica* was tested with the `dvips` written by Tomas Rokicki. His electronic mail address is `rokicki@neon.stanford.edu`. His program is available by anonymous `ftp` from `labrea.stanford.edu` as the files '`/pub/dvips*`'. TEX/*Mathematica* also works with the `dvips` available from ArborText Inc., 535 West William Street, Suite 300, Ann Arbor, Michigan 48103, telephone (313) 996-3566.

The `\special` commands are slightly different for these two `dvips` processors. Both forms are given at the end of the macro file '`mathematica.tex`' in the `\psfiginit` macro. The Rokicki form is selected in the distribution kit:

```
\def\psfiginit{%
% ArborText dvips
%\special{ps: plotfile /usr8/dan/tex/inputs/psfig.pro global}
%\special{ps: plotfile /usr8/dan/tex/inputs/mma.pro.1.2 global}
% Rokicki dvips
\special{header=/usr8/dan/tex/inputs/psfig.pro}
\special{header=/usr8/dan/tex/inputs/mma.pro.1.2}
}
```

If you use the ArborText driver, change these statements to

```
\def\psfiginit{%
% ArborText dvips
\special{ps: plotfile /usr8/dan/tex/inputs/psfig.pro global}
\special{ps: plotfile /usr8/dan/tex/inputs/mma.pro.1.2 global}
% Rokicki dvips
%\special{header=/usr8/dan/tex/inputs/psfig.pro}
%\special{header=/usr8/dan/tex/inputs/mma.pro.1.2}
}
```

Note that in either case the directory '`/usr8/dan/tex/inputs/`' will be different on your system (see Section 9.3 [Install], page 32).

Other `dvips` processors can be used with TEX/*Mathematica* by adapting the `\psfiginit` macro in '`mathematica.tex`' to their `\special` command syntax.

10.3 `math`: *Mathematica* mode Emacs package

'`math.el`' is the Emacs package that communicates with *Mathematica*. It was written by David Jacobson. '`math.el`' is included in the TEX/*Mathematica* distribution. It is also available by anonymous `ftp` from `otter.stanford.edu`; see the file '`README`' in the top-level directory there for more information. David Jacobson's electronic mail address is `jacobson@cello.hpl.hp.com`.

10.4 `psfig`: TEX PostScript graphics macros

`psfig` is the TEX macro package that allocates space for PostScript graphics and adds the appropriate `\special` commands to the '`.dvi`' file. It was written by Trevor Darrell and is available by anonymous `ftp` from `whitechapel.media.mit.edu`. Trevor Darrell's electronic mail address is `trevor@brixton.media.mit.edu`.

Index

M

P

Q

R

S

T

V

Table of Contents