Interactive TEX/*Mathematica* documents*

Dan Dill
Chemistry Department, Boston University
Boston MA 02215

December 16, 1991

Abstract

Tools are presented that provide facilities of *Mathematica* Notebooks in a UNIX environment and permit the full use of TEX in their annotation. With these TEX/*Mathematica* 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

The *Mathematica* notebook front end available on Macintosh, NeXT and IBM personal computers is an exciting step toward the goal of interactive mathematical documents.

Unfortunately, the notebook front end is not yet available under X windows and so access to *Mathematica* is more cumbersome on UNIX systems than on personal computers. This limitation is particularly significant for students, who may be new to computers and certainly to *Mathematica*. Indeed, a significant part of the ready acceptance of *Mathematica* by students is its very welcoming, easy to use notebook front end.

Further, while Notebooks allow annotation of *Mathematica* sessions, these annotations do not support TEX and so full mathematical notation cannot be easily used to describe *Mathematica*-based explorations. This restricts the effective use of *Mathematica* 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, since mathematics is central to their exposition.

The tools described here were developed to provide some of the facilities of notebooks in a UNIX environment and to permit the full use of TEX in their

*Two articles based on this work have been published in *The Mathematica Journal*, Nos. 3 and 4, 1991.

annotation. The primary tool is the GNU Emacs package `tex-mathematica` for interactive exchange between a TEX document and *Mathematica*. However, `tex-mathematica` is independent of TEX and so interactive *Mathematica* documents can be prepared without having to use TEX. This is especially appropriate for students, who should not be expected to have to learn TEX to use *Mathematica*. Their teachers, however, may well want to use TEX and macro packages are provided to format *Mathematica* text. It is hoped that students (and others), when they see what beautiful documents can be created in this way, will be encouraged to use TEX in their work with *Mathematica* too.

2 Using `tex-mathematica`

To create an interactive TEX/*Mathematica* document, use the command `M-x tex-mathematica`. This will set up `TeX/Mathematica mode` in the working buffer and startup *Mathematica* as a shell process in `Mathematica mode` [1] in a second buffer. The command `C-c o` switches to the *Mathematica* buffer in another window.

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

Mathematica code in a TEX document is contained in "active 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:

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

The enclosing rules delineate cells in the TEX-formatted document. Here is the cell after `C-c u` (update):

```
Apply[Plus, Array[{{#,#^3},{#^2,1}}&, 100]] // MatrixForm
.
Out[14]//MatrixForm= 5050       25502500

                     338350     100
```

The output is separated from the input by a "." line (this can be changed), and contains the output marker `Out[`..., if any.

An important feature of *Mathematica* notebook frontends is the ability to specify that certain active 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; the first form lets an initialization cell be bypassed optionally. An initialization cell is marked with $\boxed{\mathrm{I}}$ at the right edge of the cell start rule,

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

Note that these particular commands generate no *Mathematica* output and so there is no output portion to the cell.

Mathematica text may be set off with `TAB` characters; these are 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 over several lines. Multiple statements must be separated by '`;`' and when a given statement extends over several lines, any lines that could be interpreted by *Mathematica* as a valid (but premature) 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[13]= {{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.

David Jacobson's GNU Emacs package `math` [1] contains several aids that have been made available in the TEX 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`), 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 '\'. Since `tex-mathematica` uses `Mathematica mode`, the help, name completion and `.m`-file syntax error location commands also work directly from the *Mathematica* buffer.

Note that `tex-mathematica` and `math` are complimentary approaches to interacting with *Mathematica*. With `tex-mathematica` the emphasis is on interactive document preparation and so editing can be done in the TEX/*Mathematica* buffer. With `math` the emphasis is on the *Mathematica* session itself and so one interacts directly with *Mathematica* in the *Mathematica* buffer. Because `tex-mathematica` uses `math`, both approaches can be used as appropriate.

A full on-line description of the facilities of TEX/*Mathematica*, using the GNU Emacs Info documentation browsing system, is given by the command `C-c h`. A summary of the commands available in the TEX/*Mathematica* and *Mathematica* buffers is given by the command `C-h m` when in each buffer. The version of TEX/*Mathematica* that is running is displayed with the command `C-c v`.

3 TEX/LATEX documents

One reason for creating this package was so that students would be able to edit, annotate and save their work with *Mathematica*. This is totally independent of the use of TEX. The cell creation, send, update and replace commands described here, and the help and syntax error facilities, work irrespective of what typesetting tool, if any, is used for text. For convenience, however, we will assume that the text buffer uses TEX.

Macros are provided for formatting of *Mathematica* cells by TEX. In 10-point TEX documents the way to input these macros is 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. 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

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

The first of these rules can be illustrated explicitly as follows. First, the Clebsch-Gordan coefficients are given by the function[1]

```
ClebschGordan[{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_] := ClebschGordan[{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.

[1] Determining the argument list of `ClebschGordan` is an example of the convenience of the help facility. It was gotten by using `C-h e` and then pasting in the text displayed in the `*Help*` buffer.

4 *Mathematica* graphics

Graphics generated by *Mathematica* can be incorporated into TEX documents using Trevor Darrell's `psfig` TEX/LATEX macros [2]. These are described fully in the detailed documentation provided as part of the `psfig` kit. (The `epsf` macros of Tomas Rokicki's program `dvips` [3] can be used instead.) Including graphics in a TEX/*Mathematica* document is a three-part process. First the `psfig` macro must be read in and initialized, then graphics are generated to files with appropriate PostScript bounding box information, and finally files are placed in the document.

Before *Mathematica* graphics files can be included in a TEX document, the `psfig` macro must be set up. For a TEX document use

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

For a LATEX document use

```
\documentstyle[mathematica,psfig,...
...
\begin{document}
...
\psfiginit
```

The command `\psfiginit` must come before any use of `\psfig` (see below); in LATEX documents `\psfiginit` also must come after `\begin{document}`.

Figures are incorporated via PostScript graphics files that contain the proper bounding box information. The *Mathematica* command `PSTeX[-graphics-, "file"]`, defined in `PSTeX.m`, generates these graphics files. If `PSTeX.m` has not yet been read in (say by the *Mathematica* initialization file `init.m`), it can be loaded with

```
<<PSTeX.m
```

To save space, `PSTeX` does not include in the file the PostScript prolog containing *Mathematica*-specific definitions. This means that this prolog file must be downloaded to the printer before the file can be printed. This is done by the `dvips` processor, using `\special` commands inserted in the `.dvi` file by `\psfiginit`.

The figure is scaled to a height equal to `PSTeXHeight` points (72 points equal 1 inch), using a scheme developed by Cameron Smith [4]; the default height is 100 points. After the figure has been stored, the height can still be changed setting the `height` variable of `\psfig`. However, if the height is determined by `PSTeXHeight`, the font size for labels is 7 points (or whatever the value of the

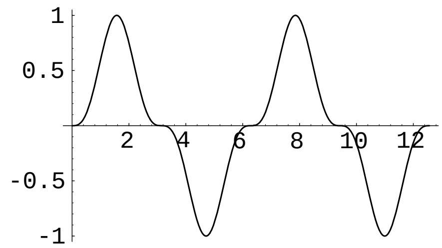

Figure 1: *Mathematica* plot of $\sin^3 x$.

variable `$DefaultFont` is); if the height is determined by the `height` variable, the whole figure is scaled, including the fonts.

Notice that a file generated by `PSTeX` generally should not be printed directly, because it does not incorporate the PostScript prolog file, because the figure is scaled in the way described above, and because the bounding box adjustment shifts the figure to the lower left corner of the page. The command `PSFile`, defined in `PSFile.m`, produces a graphic in a file, centered and scaled to fill the whole page.

The following example shows how to generate a figure to a file:

```
fig = Plot[Sin[x]^3, {x, 0, 4Pi}]; \
        $DefaultFont = {"Courier", 9.}; \
        PSTeX[fig, "sin3x"]
.
PSTeX::file: Graphics being processed (without prolog) to file "sin3x.ps".

Out[10]= -Graphics-
```

Use `ESC ` "` (`M-x quoted-insert "`) to insert a double quote in a TEX buffer. Notice that the suffix `.ps` is added automatically to the file name, and that the font size was changed from the default (7 points).

Finally, the figure can be placed in a LATEX document with commands like

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

which produce the figure seen here. A macro is provided with TEX/*Mathematica* to place figures in a TEX document in a similar way.

5 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 file `README` there describes what to do. I can be reached at Internet address `dan@chem.bu.edu`.

6 Acknowledgment

I am grateful to Cameron Smith for providing me with the details of his scheme to adjust the bounding box in *Mathematica* graphics for use with `psfig/TeX`.

References

[1] David M. Jacobson. *Mathematica* Mode for Gnu Emacs. *Mathematica J.*, 1(2):29, 1990.

[2] Trevor J. Darrell. Incorporating PostScript and Macintosh figures in TeX. Describes the `psfig`/TeX macro package. `psfig`/TeX can obtained by anonymous `ftp` from `whitechapel.media.mit.edu`, directory `/psfig`.

[3] Tomas Rokicki. Dvips: A TeX driver. Describes this `dvi`-to-PostScript driver and its `epsf` macros to include PostScript graphics into a TeX file. It is available by anonymous `ftp` from `labrea.stanford.edu`, directory `/pub`.

[4] Cameron Smith. Fixing the bounding box in *Mathematica* graphics. Cameron Smith will describe this work in a forthcoming article in *The Mathematica Journal*.