MathPSfrag 2.0 Manual

Johannes Große

Institute of Physics, Jagellonian University,
Reymonta 4, 30-059 Cracow, Poland.

Abstract

This is the complete manual of MathPSfrag, a package for the creation of LaTeX labels for MATHEMATICA plots. The manual is not meant as an introduction but rather as a documentation of all available commands. An overview article may be found in [1]; for a first impression have a look at the presentation in the `MathPSfrag-2.0/latex/beamer-example-eurobachotex-2007/` directory. In case you have taken a look at either already, you might want to skip the introduction.

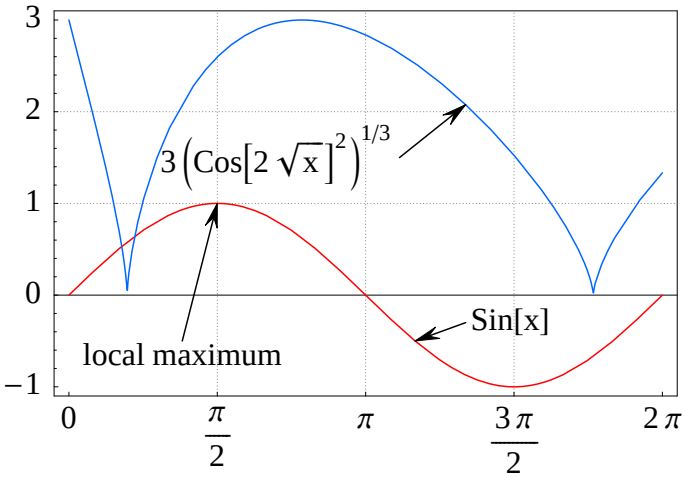


Fig. 1a) MATHEMATICA

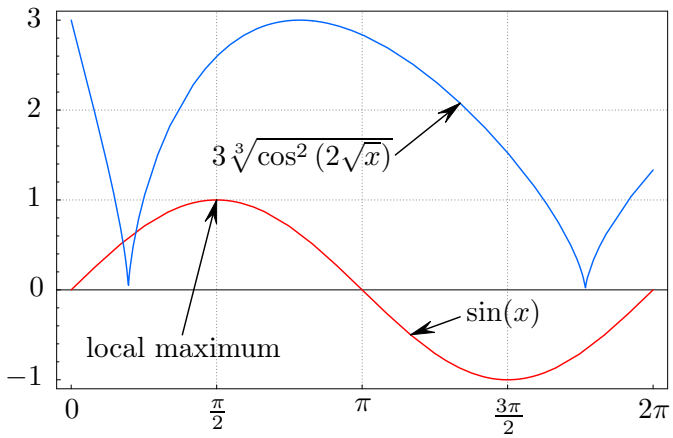


Fig. 1b) MathPSfrag

Email: `grosse@th.if.uj.edu.pl`
http://wwwth.mppmu.mpg.de/members/jgrosse/mathpsfrag

Contents

1 Introduction

Many programs producing EPS graphics do not allow the inclusion of LaTeX commands. While there exist several solutions to work around these difficulties, they all have various drawbacks. (See [2] for a discussion of several methods and an alternative approach to overcome these difficulties.) In this article, we will focus on a particular existing solution, the PSfrag package [3], which provides LaTeX macros allowing to replace pieces of text ("tags") in an EPS file by an arbitrary LaTeX construct.

However, for PSfrag to work, the application must write tags unaltered into the EPS file. For Mathematica [4, 5], this requirement amounts to using single words, strictly consisting of alphanumeric characters only. As a consequence, the user has to work most of the time with an inconveniently labeled graphic and is furthermore required to keep track of the tags used in the substitution macros.

On the other hand, it is not always possible to use Mathematica's conventional export function as it produces EPS files requiring the inclusion of additional fonts into the document. This means configuring the local TeX installation such that it finds the fonts provided by Wolfram Inc. [6, 7, 8], a process often not being under the author's control. A way out is to include the fonts into the EPS file and set the font family to a standard PostScript one.

```
Plot[..., TextStyle→{FontFamily→"Times"}]
Export[..., ConversionOptions→{"IncludeSpecialFonts"→True}]
```

However, automatic inclusion of Mathematica's special fonts, which irrespective of the chosen `FontFamily` are used for displaying important symbols like brackets, is only a last resort. While the slight mismatch between a standard PostScript font's appearance, (here Times Roman, cf. fig. 1a), and that of LaTeX's standard font (Computer Modern) may be acceptable in case of ordinary text labels, mathematical expressions like square roots or fractions cannot compete with LaTeX's typesetting quality in this approach.

Font inclusion is also a feature that—like the visually displeasing option to not use special fonts at all—has become available only starting from Mathematica version 4.2.1. Consequently, some authors simply restrict labeling of Mathematica plots to a bare minimum.

MathPSfrag [9] is a package that conveniently produces publication-quality labels in EPS files generated by Mathematica. MathPSfrag automates many (often all) tedious details related to the use of the standard LaTeX package PSfrag, while still allowing manual fine tuning. As a demonstration of the degree of automation, compare fig. 1a, which has been generated by using the standard Mathematica command `Export`, and fig. 1b, generated by MathPSfrag's export instruction.

While the solution presented here, relies on the PSfrag package, it avoids many of its shortcomings by providing a semi-automatic layer. In many cases it is sufficient to simply use the new `PSfragExport` command.

MathPSfrag also allows the creation of stand-alone images that do not need any additional preparations in the manuscript. This is achieved by calling LaTeX and Ghostscript from within Mathematica requiring the user to have a full LaTeX distribution on the same machine where Mathematica resides.

2 Naming conventions/Typography

Thoughout this document, file names are denoted Unix-style (`path/to/files`) unless they are system-specific. The corresponding Windows path would be `path\to\files`. Similarly, "folders" are called directories. The basis directory of the MathPSfrag package is called `MathPSfrag-2.0/`, but the user may rename it as she pleases.

The basis directory of the TEX installation will generically be called `{TEXMF}/`. The real location might be for example. `/usr/share/texmf/`, `/usr/local/texmf/` or `C:\` ↩`localtexmf\texmf\`.

Command line quoting is performed with single quotes `'` where appropriate in the Unix-context. For Windows these should always be double quotes `"`.

Line breaks that are not to be entered (but are rather a consequence of limited space in the manuscript), are marked with ↩ at the beginning of the next line. (As the reader might have observed in one of the paragraphs above.) Should a hyphen `-` happen to be the last character of the line, it nevertheless should be entered verbatim, when a ↩ sign is encountered.

Mandatory arguments are denoted ⟨*musthave*⟩, optional arguments [*maybe*].

3 Setup

3.1 Requirements

Strictly required

- MATHEMATICA 4.0 or later
- LaTeX, `dvips`, Ghostscript
- PSfrag LaTeX package [3]

Recommended

- MATHEMATICA 5.1 or later
- CustomTicks [10]
- pst-pdf LaTeX package [11]
- pdftricks LaTeX package [12]
- BEAMER LaTeX class [13]
- `pdfcrop` [14] (which requires Perl)

Note that the ps4pdf package is deprecated by pst-pdf. Both provide a shell script of the name `ps4pdf`, which may conflict.

3.2 Installation

1. Check the requirements!
2. Unzip the MathPSfrag package
3. Read the `README.TEXT` file

4. Download CustomTicks [10] (strongly recommended) and put `CustomTicks.m` into the main directory of MathPSfrag.
   (Presumably this directory is named `MathPSfrag-2.0`.)

5. Try to translate `is-psfrag-installed.tex` with LaTeX. If it does not work, install the psfrag LaTeX package.

6. Is MATHEMATICA version 5.0 or earlier installed?
   Install

   - `MathPSfrag-2.0/latex/mma4tex/mma4tex.tex` and
   - `MathPSfrag-2.0/latex/mma4tex/mma4tex.sty`

   into subdirectory `{TEXMF}/tex/latex/tex4mma` of your TeX distribution. Need more details? Read section 9.

7. Load "MathPSfrag-Test.nb" and execute line by line. Errors if any probably occur after the `MathPSfragConfigurationTest[]` line. Make sure that LaTeX, `dvips` and Ghostscript are in the system's execution path – that's the clean solution – or find the exact location of the executables and provide them via the `UnPSfrag` options marked by `(* EDIT THIS *)` in the test notebook. Need more details? Read section 5.4.

Congratulations.

3.3 Testing

The LaTeX examples are set up in such a way that they translate even without the user having to generate her own set of graphics. However for testing, it is useful to *ensure* that only the user generated content is included, which requires some editing.

1. Check `xmpl-psfrag-orig.pdf` to get an impression of what we want to create.

2. Edit `MathPSfrag-2.0/latex/common/inc-preamble.tex`: Comment the line

   ```
   \graphicspath{{../myfigs/}{../origfigs/}}
   ```

   and uncomment the line

   ```
   \graphicspath{{../myfigs/}}
   ```

   This will ensure that only the *user's* graphics are loaded and not the ones delivered with MathPSfrag. If you have changed the default location of where the graphics are exported to, you have to change above line accordingly.

3. LaTeX the example in `MathPSfrag-2.0/latex/xmpl-psfrag/`. Hint: You might execute `Makefile.sh` (Linux or Mac) or `Makefile.bat` (for Windows).

4. Have a look at the output and compare with `xmpl-psfrag-orig.pdf`. Does it look good? Minor differences might be due to different MATHEMATICA versions. Consider to bug report in case of errors, but have a look at section 12 before.

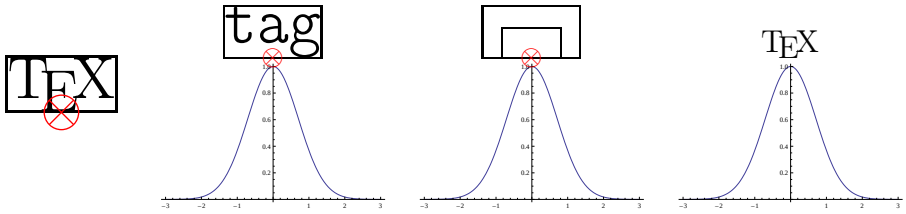


Figure 2: Action of `\psfrag{tag}[bc][bc]{\TeX}`. The first coordinate pair `[bc]` picks the reference point (red crossed circle) in the TEX expression (fig. a), the second one in the `tag` that is part of the original EPS file (fig. b). Positioning is achieved by overlaying both boxes such that both reference points coincide (fig. c). The final result is shown in fig. d. Scales are exaggerated for better illustration.

3.4 Why configuration is needed

MathPSfrag is divided into two layers: A part that produces `\psfrag` commands and correspondingly tagged EPS files, and a part that merges these two files by feeding them into LATEX. For the latter part, MathPSfrag needs to know the exact location of the LATEX, `dvips` and Ghostscript executables. Under Unix-like systems or if you have a properly configured system path under Windows (unlikely), chances are that you do not have to configure MathPSfrag at all. If for some reason there, there is no LATEX distribution on your MATHEMATICA system, you may want to use the `unpsfrag` bash-shellscript instead of the `UnPSfrag` command. It might also work in a Cygwin [15] environment, but hasn't been tested yet.

4 PSfrag LATEX package

This is intended to be a short introduction to the LATEX package PSfrag explaining only the essential features necessary to understand the corresponding MATHEMATICA package's internals and to take advantage of its manual options if automatic placement does not yield the desired result. The full documentation can be found in [3].

PSfrag provides the macro

`\psfrag{`⟨*tag*⟩`}[[`*texposition*`]][[`*psposition*`]][[`*scale*`]][[`*rot*`]]{`⟨*LATEX*⟩`}`

---

which replaces any occurrence of ⟨*tag*⟩ in the output of an EPS file by ⟨*LATEX*⟩. This process has been illustrated in fig. 2. According to [3], "all `\psfrag` calls that precede an `\includegraphics` (or equivalent) in the same or surrounding environments" will affect the output of the included graphics; i.e., `\psfrag` commands can be defined either locally, to act on strictly one graphic, or globally, thus acting on all graphics in a document.

[*texposition*] and [*psposition*] are optional arguments that allow to set (first) the vertical (top, bottom, Baseline, or center) and (second) the horizontal (left, right, center) alignment of the replacement text by specifying the respective first character of the choices given in parentheses. The arguments refer to the position of the reference point in the respective bounding boxes. The LATEX construct is placed such that its reference point is at the position of the corresponding POSTSCRIPT (tag) box' reference point, cf. fig. 3 and 2.

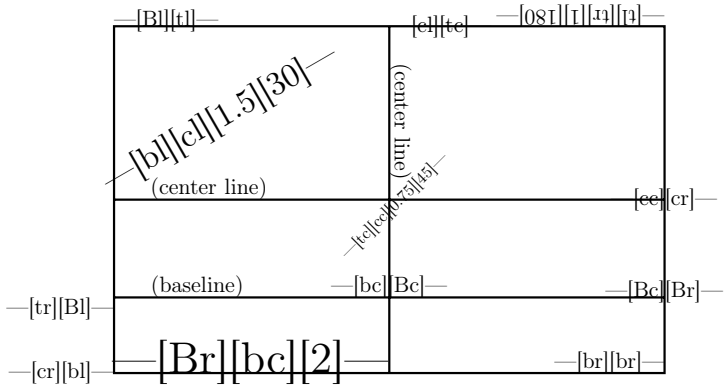


Figure 3: Illustration of the various optional arguments of the `\psfrag` command, taken from [3] with minor changes. The first option determines the alignment of the LaTeX description, while the second one is responsible for the point to which the LaTeX macro is attached.

[*scale*] and [*rot*] permit scaling and rotation of the inserted box, where the rotation (in degree) is relative to the slope of the PostScript bounding box such that a value of "0" preserves the orientation, see fig. 6. Scaling is best achieved by using LaTeX scaling commands, like `\Large`, instead of the [*scale*] option, since the standard LaTeX fonts consists of bitmaps rendered specifically for the chosen size and do not rescale well.

Since PSfrag exchanges the labels of an EPS image, these may protrude from the bounding box of the resulting image. Fortunately, `\includegraphics` offers the options `bb` and `trim`, which may be used to override or correct the obsoleted information in the EPS file. In particular, in conjunction with some of the PDF production techniques described in the section 6, correct(ed) bounding boxes can be important because protruding material might be clipped.

5 MathPSfrag interface commands

There are only three commands needed to control MathPSfrag's EPS generation: PSfrag`Export`, which supersedes Mathematica's `Export` command, and PSfrag, which allows overriding of the defaults for particular expressions. In addition the UnPSfrag command is provided which calls LaTeX and Ghostscript to carry out the PSfrag replacements and create an (ordinary) EPS and PDF image that can be included with the usual `\includegraphics` command.

5.1 PSfragExport

`PSfragExport[`⟨*basename*⟩`,` ⟨*graphics*⟩`,` [*options*]`]` converts ⟨*graphics*⟩, the usual `Graphics` construct returned by Mathematica commands like `Plot`, to an EPS file and a LaTeX file containing `\psfrag` macros. `PSfragExport` returns the ⟨*basename*⟩ and the names of the generated files in a list, suitable for processing by `UnPSfrag`; i.e. `UnPSfrag[PSfragExport[...]]`. PSfragExport

The `UnPSfrag` command, if provided, will merge these two into a single EPS file and will also produce a PDF version. More about this in section 5.3.

[*options*] can be any combination of the following options, listed with their parenthetic defaults.

- TeXSuffix→"⟨*string*⟩" ("-psfrag.tex")
- EpsSuffix→"⟨*string*⟩" ("-psfrag.eps")
- RenumberTags→⟨*boolean*⟩ (False)
- Format...→⟨*psfrag*⟩ (PSfrag[#]&)

Any other options will be passed on to `Export` or applied to the graphics using a `Show` command, respectively.

The respective file names of the LaTeX and EPS file are determined by ⟨*basename*⟩ to which the value of the options `TeXSuffix` and `EpsSuffix` is appended. Unless a base name containing a path is given, the output is written to the current directory, which can be set using MATHEMATICA's `SetDirectory`.

The option `RenumberTags→True` will renumber all tags and represent the number as one of 52 (small and capital) Latin characters or a combination of letters when the number is larger than 52. This feature, which generates very short tags, sometimes required to achieve a more precise positioning.

For completeness' sake, it shall be mentioned that there is a number of `Format...` options, which determine how the automatic processing of text generating graphics options is handled. The default is pretending the user having applied `PSfrag` without any additional options to the text content of the respective option. For example the `Plot` option `PlotLabel→"foo"` would be turned into `PlotLabel→PSfrag["foo"]` due to the `FormatPlotLabel` option having the value `PSfrag[#]&`.

The `Format...` options provide a method for changing the default treatment of particular `Plot` options. Suppose one would want to display a different number of significant digits on the x- and y-axis. This could be achieved by using

```
FormatTicksX→(PSfrag[#,TeXCommand→MyDigitsX[#]]&)
FormatTicksY→(PSfrag[#,TeXCommand→MyDigitsY[#]]&)
```

with `MyDigitsX/Y` suitable, user-defined functions that accept a number as argument and return a string containing a well formatted number. However, the recommended method of customizing axes labels is using the CustomTicks package [10]. A minimal example is given in section 11.2.

5.2 PSfrag

PSfrag

Usually one will not use the `Format...` options for the purpose of manually controlling the output, but wrap the

```
PSfrag[⟨expr⟩, [options]]
```

command directly around any MATHEMATICA expression ⟨*expr*⟩ appearing as text in a graphic, such as the argument of a `PlotLabel→...` or `AxesLabel→...` option or a `Text` graphics directive; e.g.,

```
p=Plot[...,PlotLabel→PSfrag["χ^2-test",...]];
PSfragExport["chisquare", p];
```

PSfrag processes the following options, whose defaults have been put in parentheses.

- TeXCommand→"⟨*string*⟩" (Automatic)
- PSfragTag→"⟨*string*⟩" (Automatic)
- TeXPosition→"⟨*yx*⟩" (Automatic)
- PSPosition→"⟨*yx*⟩" (CopyTeXPosition)
- PSRotation→⟨*number*⟩ (0)
- PSScaling→⟨*number*⟩ (1)

Actually, PSfragExport's automatic mechanism simply wraps PSfrag around all Text primitives using the default values above. However, manual wrapping has the advantage of allowing different options to be applied to expressions where the automatic behavior did not give satisfactory results.

TeXCommand→"⟨*string*⟩" uses ⟨*string*⟩ as the LaTeX command to appear in the final EPS graphic as a replacement of the corresponding expression ⟨*expr*⟩. If set to Automatic, the internal function GuessTeX is called, which is basically a wrapper around TeXForm that adds $ signs around math expressions.[1]

The content of PSfragTag is a unique alphanumeric tag that is used as a placeholder in the graphics. In most cases the user will not want to change the Automatic default, which corresponds to generating a valid tag from a string representation of ⟨*expr*⟩.

The remaining options are in one-to-one correspondence with those of \psfrag explained in section 4, see there; i.e., they determine the alignment of the generated LaTeX labels. The default behavior of TeXPosition is to extract the alignment of the surrounding Text command. (If there is none, it waits till the Text command is produced during export.) The default value CopyTeXPosition of PSPosition does exactly what it says; i.e., taking over the value of the TeXPosition option. The values of PSRotation and PSScaling are relative to the MATHEMATICA expressions, that is the default values will preserve both orientation and size.

MathPSfrag uses the FullGraphics command to determine the alignment values for TeXPosition and PSPosition. Unfortunately FullGraphics does not work for three-dimensional graphics, such that PSfragExport falls back to PSfragManualExport, which does not perform any alignment detection and can also be used for two-dimensional graphics if automatic processing is not desired. In these cases PSfrag has to be applied by hand to the argument of any Text directive and text producing option like PlotLabel. Moreover at least the values of TeXPosition will have to be given by the user.

5.3 UnPSfrag

With the commands described so far, standard PSfrag EPS pictures can be produced. MathPSfrag provides two additional features that require a full LaTeX run from within MATHEMATICA. First, creation of pre-rendered EPS images, which do not require the PSfrag package anymore. Such pre-rendered images will be called "unpsfraged" henceforth. Second, conversion to PDF images suitable for pdfLaTeX or raster images providing a preview of the final plot. For both features, MathPSfrag needs to know where to find

[1] To be more precise, GuessTeX has two options that allow the user to override or modify the code generated by TeXForm. These options take care of adding the required $ signs and also take into account part of the version dependence of TeXForm, cf. section 10

the executables for LaTeX, dvips and Ghostscript. Unless the binaries are in the system's execution path (as will be the case for Unix-like operating systems), this requires the user to set the location of the files explicitly. The notebook containing the code of all example plots shown in this manual also contains a comprehensive step-by-step guide of how to set up these paths permanently.

UnPSfrag

The syntax of UnPSfrag is

UnPSfrag[{⟨*basename*⟩,⟨*epsfile*⟩,⟨*texfile*⟩},[*options*]];

---

where ⟨*basename*⟩ is used to create the names of output files by appending suitable suffices like ".pdf" or ".eps". The other two mandatory parameters should point to the tagged EPS and PSfrag-LaTeX files that are to be merged. PSfragExport returns the three file names in exactly this format such that its output can be directly fed to UnPSfrag.

UnPSfrag has a number of additional options that control how images are created. Again only the most important ones shall be given here, while a complete list is given in the manual.

By default, UnPSfrag only produces an EPS and a PDF file, though it should be possible to generate other output formats by setting the UnPSfragOutputFormats option accordingly.

Furthermore UnPSfrag shows a low resolution bitmap preview of the final images. While this is extremely useful for checking if MathPSfrag has produced the desired results, it will enlarge the notebook considerably, which can be inconvenient for email exchange. Setting the PreviewDevice option to None will switch off this behavior.

When using the PSfrag option TeXCommand to specify LaTeX code, it might require additional style files. These can be included by UnPSfrag when providing the appropriate \usepackage command as a string via the TeXPreamble option.

IncludeGraphicsOptions is the most important option. The string it provides is handed over to the \includegraphics command in the LaTeX file used to perform the \psfrag replacements. The option can therefore be used to set the size of the rendered graphics; e.g., by setting its value to "width=7cm". Note however that the final bounding box will correctly fit the image *content* instead of exactly matching the specified *size*. This mismatch is due to the bounding box changing during PSfrag's replacement procedure. When an exact size is required, it can be reapplied in the manuscript, though it is better to adjust the size from within MATHEMATICA, the reason for which is explained in the next section.

5.4 Configuration

When no pre-rendered images are desired, configuration is not required. In order to use UnPSfrag however MathPSfrag needs to know where to find the executables for LaTeX, dvips and Ghostscript. The respective paths may be set by invoking

---

```
SetOptions[UnPSfrag,
  LaTeXExecutable→"⟨string⟩",
  DvipsExecutable→"⟨string⟩",
  GhostscriptExecutable→"⟨string⟩"
  ];
```

---

where the default values are `"latex"`, `"dvips"`, and `"gs"` or `"gswin32c"`, the last depending on whether the system is Windows based. These values are chosen to make MathPSfrag run "out of the box" provided that the binaries are in the system's search path. This will most likely be the case on Unix-like systems; whereas Windows users might either have to adjust their path environment variable or use absolute file names in the above options.[2]

Additionally `MathPSfragConfigurationTest[]` can be invoked to check if MathPSfrag is able to find all components. Step-by-step instructions that guide the user through the configuration are provided in the notebook `MathPSfrag-Test.nb`, which also generates all of the examples in section 11.

MathPSfrag evaluates the variable `$PostMathPSfrag` after being loaded, which allows setting up a private `init.m` file[3] that contains the configuration settings. It can also be used to make MATHEMATICA find MathPSfrag without explicit path; e.g.,

```
AppendTo[$Path, "C:\\my_path_to\\MathPSfrag"];
$PostMathPSfrag := SetOptions[UnPSfrag,...];
```

5.5 In the manuscript

`UnPSfrag`'ed images do not require any additional treatment or package beyond `graphics` or `graphicx`. Since EPS and PDF versions are created anyway, it is recommended to not provide any suffix for the file name in the `\includegraphics` command, such that the manuscript translates with both LaTeX and pdfLaTeX. It is possible to issue the common size options, but this will also change the labels, such that they will not have the fixed TeX sizes anymore. This can potentially reduce the fonts quality, in particular when TeX's standard computer modern fonts are used, which are composed of bitmaps optimized for a particular resolution. The `UnPSfrag` option `DvipsOptions` is by default set to `"-Ppdf"`, which makes `dvips` replace bitmap fonts by outline fonts on most systems, thus reducing the problem to a mere mismatch between the label's size and that of the manuscript's text. In any case it is advisable to adjust the size from within MATHEMATICA before rendering by applying `IncludeGraphicsOptions` to `UnPSfrag`.

It is also possible to follow a "pure" PSfrag approach, where all replacements are performed on the level of the manuscript. This is discussed in section 4. Before, a number of usage examples shall be given.

6 pdfLaTeX

For publication of LaTeX documents, the author often not only has to provide the final manuscript but also the sources to produce them subject to constraints set by the publisher.

Therefore, MathPSfrag essentially offers two levels of image production, a rather sophisticated "`UnPSfrag`" approach, which aims at producing pre-rendered stand-alone images as has been described above, and a classical, PSfrag centric approach. In this approach, MathPSfrag is used to produce a tagged EPS file plus a PSfrag translation file, which can convert the tags back into readable labels during the LaTeX run of the

[2] The backslash is an escape symbol and has to be doubled; e.g., `"C:\\Programs\\GS\\gswin32c.exe"`.
[3] This is a file that is automatically loaded upon start-up; see MATHEMATICA's Help Browser.

manuscript. These two files are generated by PSfragExport. (Further use of UnPSfrag would join this pair into stand-alone graphics.)

The decision which approach to use has an impact on both the process of image creation and on how to include the images and compile the manuscript. In particular, the PSfrag centric approach is less convenient and not as compatible as the UnPSfrag approach, since it relies on PostScript. However it has the advantage of producing considerably smaller images of potentially better typesetting quality as the used fonts are guaranteed to match the document's. Furthermore, the label's font size is fixed in this approach, which is of particular importance for the bitmapped Computer Modern, though modern TEX distributions have outline fonts available which scale more gracefully. Still visual consistency might suffer from differently scaled labels.

The following sections describe how to include PSfrag images into a manuscript and some ways for circumventing the PostScript-only restriction of the standard PSfrag use. MathPSfrag is accompanied by a set of example files each of which illustrates one of the strategies outlined below.

6.1 Standard PSfrag ("PostScript only")

The standard PSfrag way is to include the EPS and PSfrag file in a manner similar to the following code.

```
\begin{psfrags}
  \input{example-psfrag.tex}
  \includegraphics{example-psfrag.eps}
\end{psfrags}
```

The {psfrags} starts an empty group provided by PSfrag, whose sole purpose is making \psfrag definitions local to the following graphic.

The produced DVI file is then required to be converted to PostScript using dvips; and PDF can only be created by distilling from the PostScript version using for example Acrobat or Ghostscript (ps2pdf).

The disadvantage of the pure PostScript approach is that the advanced typesetting features of pdfLATEX are not available.

This restriction may be overcome by the either use of the ps4pdf, pst-pdf or pdftricks package.[4]

6.2 pst-pdf

ps4pdf is deprecated by pst-pdf, which follows the same strategy of using LATEX to produce a collective PDF container for images. pst-pdf is a bit more convenient as it does not require (or allow) to mark PostScript related code. A drawback of this feature is that the above mentioned quick fix for a discrepancy between the two outputs of a combined LATEX/pdfLATEX version cannot be applied anymore since any invisible frame boxes would not be included in the PDF container file.

That means that the use of pst-pdf will restrict the manuscript to translations with pdfLATEX at least in the sense that it is not possible to write a manuscript that will simultaneously have perfect bounding boxes when translated with LATEX. A major advantage of pst-pdf is that it provides the additional package option notightpage to support use

[4] I would like to thank Ross Moore for bringing these packages to my attention.

of `pdfcrop` by providing extra space around the images without requiring the use of individual `trim` options.

6.3 pst-pdf

pst-pdf extracts all images of a manuscript in an additional LaTeX run with the help of the preview package. These can then be turned into a PostScript file and distilled to a PDF that is called *image container*. During the pdfLaTeX run, PDF replacements for PostScript parts are read from the image container. Whenever content or order of the images are changed in the manuscript, the image container must be regenerated.

The required steps are automated in a script accompanying pst-pdf. Unfortunately, the predecessor package ps4pdf, which is still installed on many systems, has a similar script sharing the same name, `ps4pdf`. It is important to ensure that the correct version of the `ps4pdf` is used that refers to pst-pdf.

There is one additional catch due to the way `\psfrag` invalidates bounding boxes: pst-pdf must be loaded with the `notightpage` option and the `ps4pdf` script must be called with the `-crop` option.

Assuming `manuscript.tex` to be the main document, either of the following instructions produces the image container:

```
ps4pdf --crop manuscript.tex
or
latex manuscript.tex
dvips -Ppdf -o manuscript-crop.ps manuscript
ps2pdf -dAutoRotatePages=/None manuscript-crop.ps manuscript-crop.pdf
pdfcrop manuscript-crop.pdf manuscript-pics.pdf
```

Of course, a distiller different from `ps2pdf` may be used. Subsequently, the manuscript can be processed by pdfLaTeX in the usual manner, which will be use the pre-generated pictures stored in the `-pics.pdf` file.

The main advantage of this variant is that all images are collected in one single file, which can be very convenient for distribution, since it is much smaller than individual PDF versions of each image. This observation suggests that the collective image file is free of double copies of resources like fonts.

It is possible to arrange the preamble in an agnostic way, such that translation is possible with LaTeX and pdfLaTeX. This requires the image container to be created with `latex '\AtBeginDocument{\RequirePackage{pst-pdf}} \input{manuscript.tex}'`. Then it possible to simply not load pst-pdf during an ordinary LaTeX run, but only for pdfLaTeX. One should still ensure that pst-pdf receives the `notightpage` option. Note that the `ps4pdf` script already uses this line, so `ps4pdf -crop manuscript.tex` will work as expected. The preamble should read:

```
\PassOptionsToPackage{notightpage}{pst-pdf}
\usepackage{ifpdf}
\ifpdf\usepackage{pst-pdf}\else\fi
```

6.4 ps4pdf

It is recommended to use the package pst-pdf instead.

The predecessor of pst-pdf is ps4pdf, which requires all PostScript related material (including material in the preamble) to be wrapped in `\PSforPDF` instructions similarly to

```
\PSforPDF{%
\begin{psfrags}
  \input{example-psfrag.tex}
  \includegraphics%
    [width=8cm,trim=...]%
    {example-psfrag.eps}
\end{psfrags}
}% end of PSforPDF
```

and to include a `\usepackageeps4pdf` into the preamble. The `trim` option will be discussed later.

Assuming the main document is `manuscript.tex` the following sequence of instructions produces a PDF file containing all images (and only these) in PDF format.

```
latex manuscript.tex
dvips -Ppdf -o manuscript-pics.ps manuscript
ps2pdf manuscript-pics.ps manuscript-pics.pdf
```

Of course, a distiller different from `ps2pdf` may be used. Subsequently, the manuscript can be processed by pdfLaTeX in the usual manner without a need to recreate the picture container. pdfLaTeX will use the pre-generated pictures stored in the `-pics.pdf` file provided it has (up to the suffix) the same name as the main document. A change of the order or of the content of the PostScript material in the manuscript requires repeating above steps.

There is however the important catch, that material lying outside the bounding box of the PostScript graphics will be clipped. Since PSfrag effectively changes the graphical material in the EPS files, the bounding box will always be at least slightly, sometimes dramatically, incorrect and has to be manually corrected; e.g., by using the `trim` option of `\includegraphics` to provide some extra margin.

ps4pdf even allows to arrange the manuscript in such a way that naive translation with LaTeX or pdfLaTeX will produce the expected result, namely creation of a DVI or PDF file, while the image container can be created by giving an additional switch:

```
latex '\let\picsonly\relax\input{manuscript}'
```

This can be achieved by including in the preamble

```
\usepackage{ifpdf}
\ifx\picsonly\relax\usepackage{ps4pdf}\else%
  \ifpdf\usepackage{ps4pdf}\else%
  \usepackage[inactive]{ps4pdf}\fi\fi
```

```
\newcommand{\PSfraginclude}[2][]{
  \mbox{\PSforPDF{\whitebox{%
    \begin{psfrags}%
    \input{#2}%
    \includegraphics[#1]{#2}%
    \end{psfrags}%
    }}}%
  }
...
\PSfraginclude[width=5cm]{example-psfrag}
```

Figure 4: A customized `\includegraphics` for ps4pdf.

As an aside, the author observed a slight mismatch between the bounding boxes in the LaTeX and pdfLaTeX versions when following the "additional switch" strategy outlined above. Strange enough when investigating this phenomenon by surrounding the `\includegraphics` command by a frame box, this mismatch did not show up anymore. This suggests a possible workaround by surrounding the material with a white—i.e. invisible—frame box. For convenience the user might want to have a macro taking care of such complications, cf. fig. 4

The main advantage of this variant is that all images are collected in one single file, which can be very convenient for distribution, since it is much smaller than individual PDF versions of each image. This observation suggests that the collective image file is free of double copies of resources like fonts.

A disadvantage is the manual fine-tuning required to produce correct bounding boxes. When one is only interested in using pdfLaTeX the situation can be alleviated to some extent by setting the bounding box way too large using the `trim` option of `\include graphics` such that no material is clipped. A program like `pdfcrop` can then be used to automatically correct the bounding box for all images in the image container:

```
latex manuscript.tex
dvips -Ppdf -o manuscript-pre.ps manuscript.dvi
ps2pdf manuscript-pre.ps manuscript-pre.pdf
pdfcrop manuscript-pre.pdf manuscript-pics.pdf
```

6.5 pdftricks

pdftricks differs from the packages discussed so far in that it produces individual PDF files for PostScript based images. Originally it was meant to be used in conjunction with the popular pstricks package. pdftricks requires the the shell-escape capability (a.k.a. `\write18` capability) activated and the programs `ps2eps` and `epstopdf` to be installed and in the system's search path.

In that case a the manuscript can simply be translated by

```
pdflatex -shell-escape manuscript.tex
```

provided that images are enclosed in an `{pdfpic}` environment. There is no need for a `{psfrags}` environment anymore, because `\psfrag` definitions are already local.

preamble

```
\newbox\subfigbox
\makeatletter
\newenvironment{subfig}{%
  \def\caption##1{%
    \gdef\subcapsave{\relax##1}}%
  \let\subcapsave\@empty
  \setbox\subfigbox\hbox
  \bgroup}
  {\egroup
  \subfigure[\subcapsave]{\box\subfigbox}}
\makeatother
```

document

```
\begin{subfigure}
\begin{pdfpic}
...
\end{pdfpic}
\end{subfigure}
```

Figure 5: How to include a `{pdfpic}` environment into a subfigure.

```
\begin{pdfpic}
\input{example-psfrag.tex}
\includegraphics[width=...]{example-psfrag.eps}
\end{pdfpic}
```

The `{pdfpic}` environment is a `{verbatim}`-like environment that writes its content to a file; e.g., `manuscript-fig1.tex`, though the number is incremented for each `{pdfpic}`, which in turn is processed by a LaTeX, `dvips`, `epstopdf` sequence. Any additional packages required for this LaTeX run can be included using the `{psinputs}` environment. In our case this will usually mean

```
\usepackage{pdftricks}
\begin{psinputs}
\usepackage{psfrag,amsmath,graphicx}
\end{psinputs}
```

and possibly `mma4tex`, see section 9.

A restriction arises from the `{verbatim}` nature of `{pdfpic}`: it cannot be used as an argument to other macros. This may be an issue when used in conjunction with the subfigure package, whose documentation recommends to set up a `\subfigure`-like environment as presented in fig. 5.

When the `\write18` capability is not available, one may perform the same steps by hand or using the pst2pdf shell script that accompanies pdftricks. (The latter should work for example in a CygWin environment [15].) These steps have are

```
latex example-fig1.tex
dvips -Ppdf -o example-fig1.ps example-fig1.dvi
ps2eps -l -f example-fig1.ps
epstopdf example-fig1.eps
```

and have to be repeated for each `*-fig?.tex` file. Once the figures have been generated, one may switch off processing with `\NoProcess[1-100]`, where the maximum has to be sufficiently large.

7 Command reference

The following is a complete list of all public symbols provided by the MathPSfrag package. The order is approximately alphabetic, though options are collected below the corresponding function. For a truly alphabetic list, have a look at the index.

`BoundingBoxFromRaster[`⟨*rastergraphics*⟩`,`⟨*extramargin*⟩`]` returns the smallest rectangle around the non empty region of rastergraphics. This rectangle is enlarged by extramargin in all directions. This function is used when the `BoundingBoxDevice` (see there) is not set to `bbox`.

`BoundingBoxFrom`
`↪Raster`

`CreatePSfragRules[`⟨*expr*⟩`]` generates the TEX commands corresponding to any `PSfrag` used in ⟨*expr*⟩. This is an auxiliary function for ⟨*PSfragExport*⟩ and may be directly used for debugging or educational purposes. You may want to apply it to a `Graphics` object whose labels contain `PSfrag` commands. The usual application is `CreatePSfragRules[FullGraphics[anyplot]]`.

`CreatePSfrag`
`↪Rules`

1. `RenumberTags`: Option for `CreatePSfragRules` that can also be passed through `PSfragExport` (see there).
2. `PSfragPrologue`: Option for `CreatePSfragRules` that can also be passed through `PSfragExport` (see there).

`FetchGhostscriptDevices[`⟨*tempfilename*⟩`]` calls Ghostscript to find out which of the devices listed in `$MathPSfragGlobal`PredefGsDevices` are supported by the particular version of Ghostscript on the current system.

`Fetch`
`↪Ghostscript`
`↪Devices`

`Format...` A number of (internal) options for `HandleAutomaticPSfrag`, that can also be passed to `PSfragExport`. *This is an internal feature that is present to keep the design flexible and separate the general processing from the logic required for the treatment of particular* `Plot` *options. When you want to override the default behavior often it is easier to use `PSfrag` of `PSfragTicks`.* All `MathPSfrag`Format*` options can be used to fine-tune the output of the current automatic handling of axes labels and the like. Their respective default value is usually just PSfrag, which is wrapped around each `Text`

`Format`

element that is produced by Plot options like AxesLabel or Ticks. Since the default behavior of PSfrag is to call GuessTeX, you obtain the TEX code that is generated by this rather imperfect function. You can include you own TEX generating function by setting Format...→PSfrag[#,TeXCommand→mytexguesser[#]]&, where you obviously would have to implement mytexguesser to convert an arbitrary expression that could show up into a TEX command. For Tick marks these are usually just numbers, so you might want to use the FormatTicksX and FormatTicksY options to apply a function that sets exactly the number format you want. See also NumberToTeX, a function that could be used for this job.

The full list of these options is

- FormatAxesLabelX, FormatAxesLabelY, FormatFrameLabelEast, FormatFrameLabelEastRotated, FormatFrameLabelNorth, FormatFrameLabelSouth, FormatFrameLabelWest, FormatFrameLabelWestRotated, FormatFrameTicksEast, FormatFrameTicksNorth, FormatFrameTicksSouth, FormatFrameTicksWest, FormatPlotLabel, FormatTicksX, FormatTicksY.

The "rotated" variants are used when Rotated→True has been applied to the graphics.

GenerateTemporaryName

GenerateTemporaryName["⟨*prepend*⟩"] creates a string that can be used as a unique file name and starts with a given string "⟨*prepend*⟩".

GuessTag

GuessTag[⟨*expr*⟩] tries to create a unique string describing ⟨*expr*⟩ and entirely consisting of alpha-numerical characters. (This is called a "tag".) It is used by PSfrag when PSfragTag→Automatic is set (default).

- MaximumTagLength: Option for GuessTag. This sets the maximum length of a tag produced by GuessTag.
- MinimumTagLength: Option for GuessTag. This sets the minimum length of a tag produced by GuessTag.

GuessTeX

GuessTeX[⟨*expr*⟩] creates a string containing the TEX form of expr. It is used by PSfrag (see there) when TeXCommand→Automatic is set (default).

ReplacementHookCooperative

ReplacementHookCooperative Option for GuessTex which contains a list of shape {{priority, test, applybefore, applyafter},...} that can be used to customize the behavior of GuessTeX. Starting with the lowest value of priority, each test is called with the MATHEMATICA expression that GuessTeX is supposed to transform into a LATEX expression. Each test that returns True will cause the respective applybefore command to be applied to the MATHEMATICA expression. In addition after the default generation of LATEX code that is preformed in any case, the associated applyafter command is applied to the string containing the LATEX code.

ReplacementHookExclusive

`ReplacementHookExclusive` Option for GuessTeX which contains a list of shape `{{priority, test, command},...}` that can be used to customize the behavior of GuessTeX. Starting with the lowest value of `priority`, each `test` is called with the MATHEMATICA expression that is to be transformed to a LaTeX expression. The first test returning `True` – if any – will stop the default generation of LaTeX code. Instead the associated `command` is called and its return value is used as TeX code.

HandleAutomaticPSfrag

`HandleAutomaticPSfrag[graphics,opts]` wraps PSfrag commands around all textual elements generated by the graphics options `AxesLabel`, `Ticks`, `PlotLabel`, `FrameLabel` and `FrameTicks`. It accepts a number of options, all starting with Format, that allow to fine-tune this behavior. For example FormatTicksX has the default value PSfrag. Setting `FormatTicksX→(PSfrag[#,TeXCommand→myFormatter[#]]&)` would allow to write a function that gives better TeX output for all entries on the x axis. See also NumberToTeX.

The following are default handlers that take care of auto-wrapping PSfrag commands around the `Text` elements generated by the associated `Plot` option. Use OptionsToHandle to decide if this handler is called and the corresponding `Format...` options to fine-tune the formatting.

- `HandleAxesLabel`
- `HandleFrameLabel`
- `HandleFrameTicks`
- `HandlePlotLabel`
- `HandleTicks`

MathPSfragConfigurationTest

`MathPSfragConfigurationTest[[`*tempfile*`],[`*opts*`]]` checks the configuration of UnPSfrag (see there). [*tempfile*] is by default a (hopefully) unique name generated from `$SessionId`. It is written to the current directory. By default the test is not carried out if [*tempfile*] already exists, though you can set `ForceOverwrite→True` as an option to change that. Be careful. It may be better to simply provide a different name for `tempfile`.

NextLaTeXError

`NextLaTeXError[[`*n*`]]` displays the *n*-th error in a LaTeX run performed by UnPSfrag. When omitting the argument it shows the first error message.

OptionHandlers

`OptionHandlers`: Option for HandleAutomaticPSfrag and PSfragExport that assigns to `Plot` options different default handlers that actually carry out the formatting. Usually do not touch this option but instead use the corresponding `Format...` options to change the behavior or switch automatic handling off by setting OptionsToHandle accordingly.

OptionsToHandle

OptionsToHandle Option for HandleAutomaticPSfrag and PSfragExport that contains a list of all options that are taken care off by the automatics. Currently automatics are implemented for AxesLabel, Ticks, PlotLabel, FrameLabel, FrameTicks, which is the default value of OptionsToHandle. You can switch off the automatics entirely by setting this option to . The formatting of a Plot's elements can be changed by setting the corresponding Format... options.

PSfrag

PSfrag[⟨*expr*⟩, [*opts*]] can be used to achieve a higher level of control over the output of PSfragExport by attaching individual typesetting information to ⟨*expr*⟩. Use PSfrag[expr] where ever you want to display StandardForm[⟨*expr*⟩] (or Traditional ↪Form) on the *screen*, while a corresponding TEX command will be prepared for use within LATEX. A typical use is Plot[..., PlotLabel→PSfrag[expr,opts]]. The most important option to PSfrag is TeXCommand, which allows to define how expr is typeset when exported to an EPS file.

Use PSfragExport[⟨*filename*⟩, ⟨*graphics*⟩] to write an EPS file in the current working directory (see SetDirectory). PSfragExport will append a suffix (default: -psfrag.eps) to ⟨*filename*⟩. This file will contain a substitute alphanumeric tag for every PSfrag command which in turn is replaced by a TEX command (by default guessed by a stupid algorithm) using the \psfrag command (see documentation of psfrag.sty). The replacement TEX rules are by default written to filename-psfrag.tex. You may independently generate and display these \psfrag commands by CreatePSfragRules[graph ↪ics].

PSfrag is automatically threaded over a List when the argument of TeXCommand is a List of same length.

The available options are TeXCommand, TeXPosition, PSPosition, PSRotation, PS ↪Scaling, PSfragTag. Each has a separate help text. You might also want to study the documentation shipped with psfrag.sty.

PSfragTag

PSfragTag is an option of PSfrag that may be used to manually set the tag that is used by the TEX command \psfrag to substitute for a TEX command. Usually the default setting (PSfragTag→Automatic) gives good results an should be left unchanged. If you have problems with too large tags, you might also try to use RenumberTags→True as an option for PSfragExport, which yields shorter tags.

PSPosition

PSPosition is an option of PSfrag defining the alignment of the postscript text output by Export. Its default CopyTeXPosition takes over value of the TeXPosition option, while the value Automatic is replaced by the alignment of the first surrounding Text graphics directive. Explicit values can be the same as those of Position.

CopyTeXPosition

CopyTeXPosition: Default value of PSfrag's option PSPosition.

PSRotation

PSRotation is an option of PSfrag. Its range is 0 to 360 degree. It may be used to

turn the text counterclockwise in the final PostScript output.

PSScaling

`PSScaling` is an option of PSfrag. It is a positive number which is used to increase the size of any TEX command's output in the final EPS file. The default value is 1, meaning no rescaling. It is recommended not to use this option but instead use TEX commands like `\small`. MathPSfrag inserts `\PFGstyle` commands in front of each TEX expression. Issuing `\newcommand{\PFGstyle}{\small}` in the preamble will therefore scale all labels in a manuscript.

TeXCommand

`TeXCommand` is an option of PSfrag. The default PSfrag[⟨*expression*⟩, TeXCommand↵→Automatic] deduces the TEX command from the given expression. For complicated expressions, this may not give satisfactory results and you may have to set them by hand using TeXCommand→"TeX command string". (Note, that the `TeXForm` output of Mathematica is version dependent.) If the argument is not a string, it is assumed to be a function that returns a suitable TEX string when applied to the Mathematica expression.

TeXPosition

`TeXPosition` is an option of PSfrag defining the alignment of the TEX text replacing the tag in the EPS output. Possible values are any two-character combination of ("B", "t", "c", "b") and ("l", "c", "r"). The default is `Automatic`. For the x label in `AxesLabel` you should use "Bl".

TeXShift

`TeXShift`→{"⟨*x*⟩","⟨*y*⟩"} is an option of PSfrag shifting the position of the TEX box by the specified amount. ⟨*x*⟩ and ⟨*y*⟩ are TEX dimensions; e.g., `3cm` or `4pt`. When a numeric value is given, the unit `pt` is assumed.

TeXShiftX

`TeXShiftX`→⟨*value*⟩ is equivalent to TeXShift→{⟨*value*⟩,0}.

TeXShiftY

`TeXShiftY`→⟨*value*⟩ is equivalent to TeXShiftY→{0,⟨*value*⟩}.

PSfragExport

`PSfragExport`[⟨*filename*⟩, ⟨*graphics*⟩, [*opts*]] writes an Encapsulated PostScript (EPS) version of graphics into a file. Filename may contain a directory, otherwise the EPS file is written into the current working directory. This EPS file is meant to be read by `\includegraphics` and processed by the PSfrag LATEX package. The TEX commands required by PSfrag are also written into a file. Note that PSfragExport automatically appends suffices defined by `EpsSuffix`→... and `TeXSuffix`→... to create the actual file names. PSfragExport first generates automatic PSfrag commands for all textual elements of a graphic (including numbers and formulas) by calling HandleAutomaticPSfrag. All options [*opts*] are passed through. This means that in particular special formatting

instructions of HandleAutomaticPSfrag can be given directly to PSfragExport. The second step is to call PSfragManualExport, which does the actual job of creating the EPS and TEX file. A list containing ⟨*filename*⟩ and the names of the generated files is returned. This list is suitable as input for UnPSfrag, such that typical usage looks like this: PSfragExport["mydemo",myplot]//UnPSfrag

- EpsSuffix
- TeXSuffix

PSfragManualExport

PSfragManualExport[⟨*filename*⟩,⟨*graphics*⟩] converts all PSfrag entries in a Graphics object into alphanumerical tags using CreatePSfragRules. The result is exported into an ordinary EPS file, while the generated \psfrag TEX commands are written into a TEX file. The respective file names are determined from the parameter ⟨*filename*⟩ and the suffix given in the PSfragExport options EpsSuffix and TeXSuffix. PSfragManualExport returns a list of the form {filename, epsfilename, texfilename}, which is suitable as input to UnPSfrag. PSfragManualExport is internally called by PSfragExport after constructing automatic PSfrag commands for all textual elements in a graphic by calling HandleAutomaticPSfrag.

PSfragTicks

PSfragTicks[⟨*tickspec*⟩,[*opts*]] maps PSfrag[#,⟨*opts*⟩]& at each label in a tick mark specification (see the Ticks option of Plot). Note that tickspec is for one axis only, so PSfragTicks has to be applied to each axis. Since PSfragExport applied to two-dimensional graphics will automatically wrap PSfrag at the correct positions, this is only needed, when you want to have additional control over the output or for three-dimensional plots.

A typical usage example:

```
Needs["CustomTicks`"];
Plot3D[..., Ticks->{
    PSfragTicks[LinTicks[xmin,xmax], TeXPosition->"Bc"],
    PSfragTicks[LinTicks[ymin,ymax], TeXPosition->"Bl"],
    PSfragTicks[LinTicks[zmin,zmax], TeXPosition->"Br"] }];
```

TextToPSfrag

TextToPSfrag[⟨*graphics*⟩] simply wraps PSfrag around all Text directives contained in ⟨*graphics*⟩. It is used by HandleAutomaticPSfrag, which in turn is called by PSfragExport.

UnPSfrag

UnPSfrag[{⟨*outputprefix*⟩,⟨*epsinfile*⟩,⟨*texinfile*⟩},[*opts*]] transforms an EPS graphics file ⟨*epsinfile*⟩ relying on \psfrag commands (contained in ⟨*texinfile*⟩) into a standalone graphics. outputprefix should give a valid filename when a suffix like .eps or .pdf is appended.

Example:

```
UnPSfrag[PSfragExport["example",myplot]];  or
PSfragExport["example",myplot]//UnPSfrag;
```

`BoundingBoxDevice` is an option for UnPSfrag that defaults to Ghostscript's `bbox` device. Do not change this option unless you have good reason to do so. The format is a *device triplet* as described for the PreviewDevice option. You may choose a raster device like `png` or `pnm` here in which case Mathematica will determine the bounding box from the bitmap graphic by counting non-empty rows and columns. This is not quite as precise as the bounding box generated from Ghostscript's `bbox` device and takes more memory and time. It has the advantage of correctly dealing with invisible elements, which are not cut off by Ghostscript's `bbox`.

BoundingBoxDevice

`DvipsExecutable` is an option for UnPSfrag that contains an absolute or relative name of the `dvips` command. On Unixes this will usually be in the system path, so no change will be required in most cases. On Windows systems `dvips` neither has a standard location nor is it necessarily in one of the folders given in the system's `PATH` environment variable. In that case, you will have to manually search the `dvips` program and set the absolute path explicitly, e.g. DvipsExecutable→`"C:\Programs\texmf\bin\dvips"`. Also make sure to set the correct values of LaTeXExecutable and GhostscriptExecutable. You can check your configuration with MathPSfragConfigurationTest. The `MathPSfrag-Test.nb` notebook gives a step-by-step guide for the correct configuration of UnPSfrag.

DvipsExecutable

`DvipsOptions` is an option for UnPSfrag that is passed on to `dvips` when producing EPS files. Its default value `"-Ppdf"` will ensure replace bitmap by outline fonts on most modern systems.

DvipsOptions

`ForceOverwrite`→`boolean` Option for UnPSfrag and for MathPSfragConfigurationTest (see there, respectively).

ForceOverwrite

`GhostscriptExecutable`→`"`⟨*string*⟩`"` is an option for UnPSfrag that contains an absolute or relative name of the Ghostscript command. On Unixes this defaults to `gs` and will usually be in the system path, so no change will be required in most cases. Sometimes you have to change this to `gs-gpl`. On Windows systems, this option defaults to `gswin32c`, but the program neither has a standard location nor is it necessarily in one of the folders given in the system's `PATH` environment variable. In that case, you will have to manually search the `gswin32c` program and set the absolute path explicitly; e.g. GhostscriptExecutable→`"C:\Programs\texmf\bin\latex"`. Also make sure to set the correct values of LaTeXExecutable and DvipsExecutable. You can check your configuration with MathPSfragConfigurationTest. The `MathPSfrag-Test.nb` notebook gives a step-by-step guide for the correct configuration of UnPSfrag. (Also have a look

GhostscriptExecutable

at $PostMathPSfrag explained there.)

IncludeGraphicsOptions

IncludeGraphicsOptions is an option for UnPSfrag that can be used to change the size of the generated image. For example IncludeGraphicsOptions→"width=3in" will change the size of the plot. However the size is only approximate since LaTeX does not include the size of the labels in the calculation.

LaTeXExecutable

LaTeXExecutable is an option for UnPSfrag that contains an absolute or relative name of the latex command. On Unixes this will usually be in the system path, so no change will be required in most cases. On Windows systems latex neither has a standard location nor is it necessarily in one of the folders given in the system's PATH environment variable. In that case, you will have to manually search the latex program and set the absolute path explicitly, e.g. LaTeXExecutable→"C:\Programs\texmf\bin\latex". Also make sure to set the correct values of DvipsExecutable and GhostscriptExecutable. You can check your configuration with MathPSfragConfigurationTest. The MathPSfrag-Test.nb notebook gives a step-by-step guide for the correct configuration of UnPSfrag. Also have a look at $PostMathPSfrag.

LaTeXOptions

LaTeXOptions is an option for UnPSfrag that contains command line options passed on to LaTeX during the creation of EPS files.

MasterTeXFile

MasterTeXFile is an option for UnPSfrag that contains the full TeX file that is used to convert the psfrag-EPS into a final PostScript version. Then Ghostscript is used to convert this pure PostScript file into other formats.

PreviewDevice

PreviewDevice is an option for UnPSfrag. Unless there are severe problems, the only value you might choose instead of the default is None to suppress generation of preview images. Allowed values are None or a device triplet; i.e. a list {gsdevice, suffix, options}, where gsdevice is the name of a Ghostscript device, suffix is a string that is a suitable suffix for the generated file, e.g. .eps for an EPS file, and options are additional command line options to Ghostscript that can be used to change anti-aliasing settings or the like. Typically a preview device should be a raster device, like pnm or png. Further information can be found in the documentation of the Ghostscript program.

RemoveTemporaries

RemoveTemporaries is an option for UnPSfrag that can be set to False for debugging.

SuppressOutput

SuppressOutput is an option for UnPSfrag that contains instructions to discard output of external programs.

`TemporaryDirectory` is an option for UnPSfrag that gives the name of a subdirectory where the LaTeX run in performed. All intermediate files are stored there. The temporary directory is removed after successful execution unless RemoveTemporaries has been set to False. TemporaryDirectory

`TeXPreamble` is an option for UnPSfrag that in particular can be used to change the fonts used in the image. UnPSfrag needs a TeX file to convert a psfrag-EPS file, as produced by PSfragExport, to an EPS file independent of any psfrag commands. TeXPreamble is a string that is included in the TeXPreamble of that file and can be used to include additional LaTeX packages. For example: `TeXPreamble→"\usepackage{euler}"` to change the math fonts. TeXPreamble

`UnPSfragOutputFormats` is an option for UnPSfrag that contains a list of devices triplets; i.e. `{{gsdev1, suffix1, opts1}, ..., {gsdevN, suffixN, optsN}}`. `gsdev` is a string containing the Ghostscript device like `png`, `pdf` and so forth. The `suffix` is for the filename and should contain a leading dot; i.e. `.png`, `.pdf`. The options in `opts` are passed to the respective Ghostscript call and allow to set additional options. In particular the resolution and the anti-aliasing settings. UnPSfragOutputFormats

8 General auxiliary functions

`PatchType1FontIntoEps[`⟨*infile*⟩`,`⟨*outfile*⟩`]` copies missing fonts into an EPS file. This is a hack that will result in possibly usable but not standard conformant EPS files. The upshot: **Do not use this.** You will not need this when you replace all expressions with MathPSfrag, which is a clean solution. PatchType1FontIntoEps

`NumberToTeX[`⟨*number*⟩`,[`*opts*`]]` returns a string containing TeX code that represents number. There are several options that influence the appearance of number: IntegerPartDigits determines the number of digits before the decimal point in a floating point number, FractionalPartDigits sets the number of digits after the decimal point. When ChopExponent is `True`, exponents of the form 10^0 are suppressed. ChopImaginaryPart→`True` turns complex numbers with almost vanishing imaginary part into real numbers. In a similar manner Integerize→`True` will turn floating point numbers like `1.` into integers. ExponentSymbol→`"\cdot 10^"` will format floats in the form `1.23 \cdot 10^{4}`. This is the default behavior. Example: `NumberToTeX[Exp[10.]]`. NumberToTeX

- ChopExponent, ChopFractionalPart, ChopImaginaryPart, ExponentSymbol, FractionalPartDigits, Integerize, IntegerPartDigits, IntegerToReal, RationalToReal

`NumToBase52[`⟨*integer*⟩`]` converts a non-negative ⟨*integer*⟩ into an alphabetic string. NumToBase52

LatinCharacterSet

LatinCharacterSet is a list containing all 52 letters from "A" to "Z" and "a" to "z". Used by LatinQ.

LatinNumCharacterSet

LatinNumCharacterSetis a list of all 52 Latin letters and 10 digits. Used by LatinNumQ

LatinNumQ

LatinNumQ["⟨*string*⟩"] returns True if string consists entirely of alphanumerical characters (A-Z, a-z, 0-9).

LatinQ

LatinQ["⟨*string*⟩"] returns True if string consists entirely of letters A to Z and a to z.

9 Mathematica 4.x–5.0

MathPSfrag creates the LaTeX code that corresponds to a particular Mathematica expression by calling TeXForm. Starting from version 5.1 the TeXForm output is self-supporting code, that only requires a few standard packages. Pre-5.1 TeXForm output will require an additional style file providing Mathematica fonts that have to be configured accordingly [6, 7, 8]. In the new approach, since a large number of Mathematica symbols do not have counterparts in standard LaTeX packages, some symbols are created by gluing together existing symbols. When applied carefully, this method can achieve acceptable results; though the current implementation is not optimal yet.

The problem with the pre-5.1 output in conjunction with the PSfrag method is that it requires the Mathematica fonts to be available in the publisher's LaTeX installation, while in general they will—if at all—only be available to the author. (This should not be a problem when the author decides to follow the UnPSfrag approach, where all required fonts are embedded into the image files. It is then however necessary to include the style files by setting the according UnPSfrag option: TeXPreamble→"\usepackage{...}".)

To avoid these problems in the PSfrag approach requires a compatibility package (mma4tex.tex) that is included in the MathPSfrag distribution. Basically it parses Mathematica 4.x–5.0 LaTeX code and replaces it by code similar to the output of version 5.1/5.2. When MathPSfrag detects Mathematica 5.0 or earlier, PSfragExport creates automatically PSfrag files that attempt to load mma4tex.tex and issue a warning when unsuccessful. There is also a corresponding style file mma4tex.sty that additionally loads the required font packages and ensures that mma4tex.tex is only loaded once. It is recommended to include it in the preamble. Therefore both files have to be installed where LaTeX can find them; e.g., in a directory tex/latex/mma4tex/ relative to the texmf directory of the LaTeX distribution. Thereafter the file database ("ls-R") has to be updated. Again the details depend on the distribution. Typical names of the update command are mktexlsr or texhash. For MiKTeX there is the choice between the command line program (initexmf -u) and a graphical configuration program ("MiKTeX Options" or "Settings"; press the button "General/File Name Database/Refresh Now").

10 Known Bugs and Limitations

MathPSfrag relies on three MATHEMATICA commands: `TeXForm`, `FullGraphics` and `AbsoluteOptions`. All are potential sources of failure.

For the first one, this is due to substantial changes concerning its output in its version history, which do not seem to have been publicly documented. With the latest steps in the transition towards `amsmath` compatible LaTeX though, `TeXForm` will probably stabilize with respect to its need for non-standard LaTeX packages. To make the `TeXForm` output of MATHEMATICA versions 4.x–5.0 rely on standard packages only, a compatibility layer has been written, cf. section 9. However it will not produce satisfactory output regarding those MATHEMATICA symbols that have no direct counterpart in LaTeX. Since for most purposes the symbols available in LaTeX are sufficient, this is only a minor restriction, which also applies to the `TeXForm` output of MATHEMATICA version 5 anyway. In any case, single occurrences of malformed `TeXForm` output may easily be overridden using the `TeXCommand` option of PSfrag.

The automatic positioning relies on `FullGraphics`, which is not faithful in the sense that `Show[...]` and `Show[FullGraphics[...]]` do not yield exactly the same output. The same is true for `AbsoluteOptions` and may be a potential source of errors. The code has been written such that the `FullGraphics` plot is only used for alignment information but discarded afterward in order to isolate this discrepancy.

For `AbsoluteOptions` this has not been possible since its output is directly needed. So far the only observed difference is that `AbsoluteOptions` turns integers into floating point numbers. Therefore a mechanism to override this behavior has been implemented in `GuessTeX`, the function implementing automatic generation of LaTeX code in MathPSfrag: Floating point numbers very close to integers are converted back to integers. Clearly this introduces another potential source of mysterious errors. Therefore this mechanism can be completely configured.

GuessTeX provides two options: ReplacementHookCooperative and ReplacementHookExclusive. The former contains of a set of rules, each of which is a list of the form `{rank, test, trafoexpr, trafotex}`. The rank is simply used to determine the order of the rules (low rank first). For all rules, `test` is a function that evaluates the MATHEMATICA expression to be turned into LaTeX and determines if the rule is activated. For all activated rule, first all `trafoexpr` functions are applied to the MATHEMATICA expression, it is then turned into a LaTeX string using `TeXForm` and finally the `trafotex` transformations are applied to the string.

Currently there is only one rule (albeit in two different versions) for ReplacementHookCooperative:

```
ReplacementHookCooperative-> {
  If[$VersionNumber >= 5.1,
    {10000, True &, Identity, StringJoin["$", #, "$"] &},
    {10000, (Head[#] =!= String)&, Identity, StringJoin["$", #, "$"] &}]};
```

MATHEMATICA 5.1 `TeXForm` always returns expressions in math mode, so the default rule is to always wrap dollar sign around the final expression. MATHEMATICA pre-5.1 `TeXForm` will return expressions that are strings in text mode, such that they don't need to be wrapped in dollar signs. (This is checked by the test in the last line.)

The ReplacementHookExclusive option contains a set of rules that have the form `{rank, test, trafo}`. The main difference is that the first rule that is activated (`test`

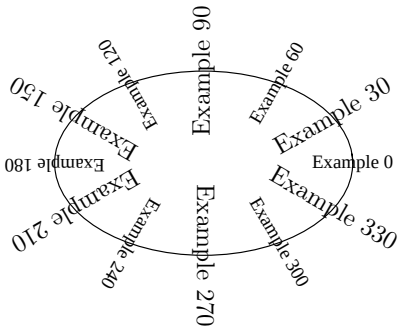


Figure 6: Example for substituting rotated text. To demonstrate that the new export function can preserve the orientation, only half of the labels have been substituted by LaTeX.

returns True) will immediately stop further evaluation by GuessTeX: The expression is handed over to `trafo` and its result is written unmodified into the PSfrag replacement file. Therefore GuessTeX expects this result to be a string. Only when none of these rules were activated, the "cooperative" mechanism described first will be employed.

The current content of the "exclusive" option is:

```
ReplacementHookExclusive->Join[{
  { 900, (NumberQ[#] && Chop[# - Round[#]] === 0 )&,
         "$" <> ToString[Round[#]] <> "$" &},
  {1000, (NumberQ[#] && (StringPosition[ToString[#], "\n"] === {})) &,
         "$" <> ToString[#] <> "$" &},
  {1100, NumberQ, "$" <> NumberToTeX[#] <> "$" &} },
  If[$VersionNumber < 5.1, {
    {5000, (Head[#] === PaddedForm) &, ToString[" " <> ToString[#], TeXForm] &}
  }, {}] ];
```

The first line undoes said `AbsoluteOptions` bug by rounding real numbers to integers when they are very close to integers already (less then 10^{-10}).

The `TeXForm` output of Mathematica up to 5.2 always turns a number's `InputForm` representation into TeX. The second line uses `ToString`, which preserves the chosen form of the number. However, this approach will not work when a floating point number is processed that has a large exponent. (The test rejects multiple line output.) The third line will process any remaining numbers by applying NumberToTeX, a simple formatting function provided by MathPSfrag.

The last line intercepts another Mathematica 4.x–5.0 peculiarity: `TeXForm` treats `PaddedForm` as a mathematical function whose name is output verbatim. `ToString` shows the correct behavior, that is merely a number. It should be noted that there is a number of other `...Form` commands that might not be treated correctly in Mathematica pre-5.1 and require similar work-arounds.

11 Figure Source Code

The source code of all example plots is listed and commented below. It can also be found in the test notebook (`MathPSfragTest.nb`) that comes with MathPSfrag and can be used to configure and check the installation of MathPSfrag. In the example code below, it

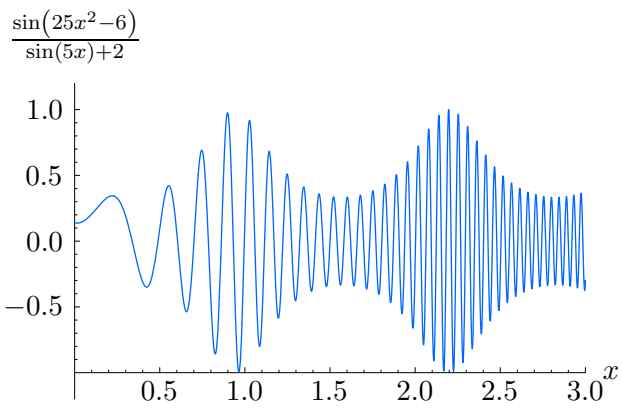


Figure 7: `HoldForm` example: Without `HoldForm`, MATHEMATICA would normal order the label on the y axis to $-\frac{\sin 6-25x^2}{2+\sin(5x)}$. The CustomTicks package has been used to avoid the typical stripped decimal "1." on the x axis.

will be assumed that all necessary packages have already been loaded. In particular the comparison of the automatic and manual examples should be educational.

11.1 Rotated Text

While in some environments MATHEMATICA does not rotate the letters of a rotated `Text` on screen, both the conventional `Export` and `PSfragExport` do the right thing, cf. fig. 6. Furthermore, it is demonstrated, that `PSfragExport` can apply the option `PlotRange→All` to the graphics before carrying out the export.

```
Show[Graphics[{
        Table[Text["Example " <> ToString[Round[phi*180/Pi]],
                   {Cos[phi], Sin[phi]}, {0, 0}, {Cos[phi], Sin[phi]}],
              {phi, 0, 2Pi - 0.01, 2Pi/6}],
        Table[Text[PSfrag["Example " <> ToString[Round[phi*180/Pi]]],
                   {Cos[phi], Sin[phi]}, {0, 0}, {Cos[phi], Sin[phi]}],
              {phi, 2Pi/12, 2Pi - 0.01, 2Pi/6}],
        Circle[{0, 0}, 1]
    }]];
PSfragExport["ex_rot", %] // UnPSfrag;
```

11.2 HoldForm plus CustomTicks

Fig. 7 is a plain example demonstrating that `HoldForm` can be used to fix the shape of an expression while `LinTicks` from CustomTicks [10] can be used to circumvent the usual stripped decimal "1." and print a much nicer "1.0" instead.

```
holdformexpr := HoldForm[Sin[25x^2 - 6]/(Sin[5x] + 2)];
holdformex = Plot[Evaluate[ReleaseHold[holdformexpr]], {x, 0, 3},
    AxesLabel→{x, holdformexpr},
    Ticks→{LinTicks[0, 3], LinTicks[-1, 1]},
    PlotStyle→Hue[0.6],
    PlotRange→{{0, 3}, {-1.2, 1.2}}, PlotPoints→100, AxesOrigin→{0, -1}]
PSfragExport["ex_hold", holdformex] // UnPSfrag
```

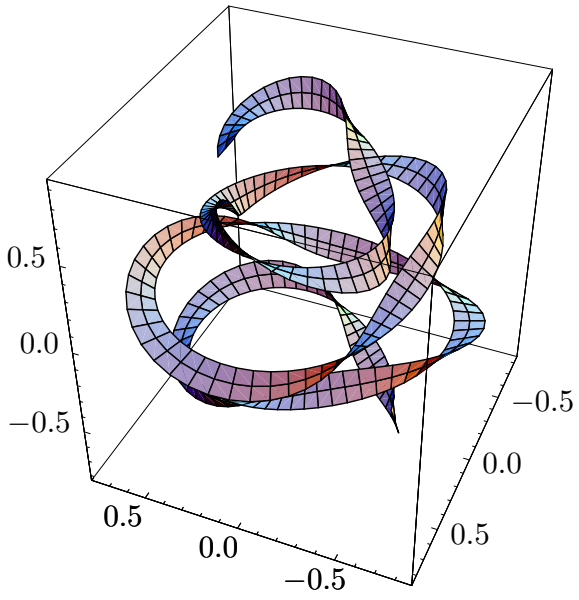


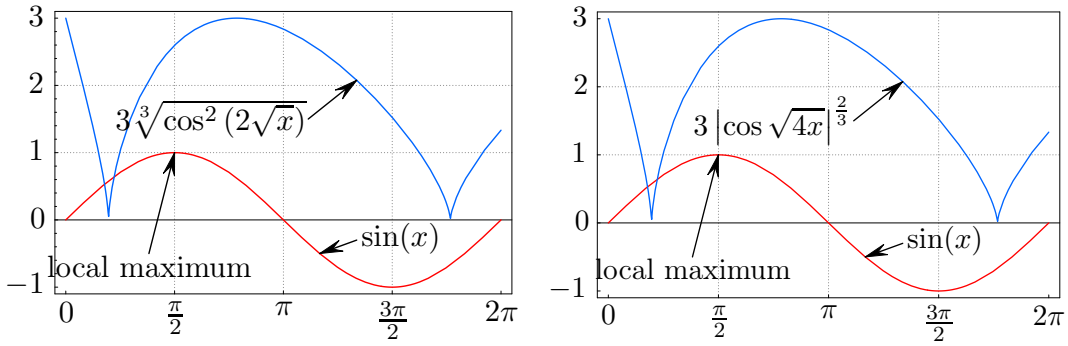
Figure 8: Three dimensional example: As there exists no `FullGraphics3D` command, manual labeling was required and the `RenumberTags` option of `PSfragExport` was used to produce smaller tags, increasing positioning precision.

11.3 Three-dimensional Knot

Here, the three-dimensional knot in fig. 8 is generated.

```
knot3d[r_, phi_] =
    Flatten[{(0.5 + 0.2*Cos[phi/5] + r*Sin[1.7phi]){Cos[phi], Sin[phi]},
        phi/(5Pi) + r*Cos[1.7 phi]}];

ParametricPlot3D[Evaluate[knot3d[r, phi]], {phi, -3Pi, 4Pi}, {r, 0.05, 0.2},
  Ticks → {
      PSfragTicks[LinTicks[-1, 1], TeXPosition → "Bc"],
      PSfragTicks[LinTicks[-1, 1], TeXPosition → "Bl"],
      PSfragTicks[LinTicks[-1, 1], TeXPosition → "Br"]},
  PlotPoints → {200, 3}, ViewPoint → {-1.1, 2.6, 2.2}]
PSfragExport["ex_3d"] // UnPSfrag;
```



a) automatic b) manual

Figure 9: Comparison: a) Plot exported with `PSfragExport` vs. b) Plot without resorting to automatics; i.e., exported with `PSfragManualExport`. Additionally, the typesetting of the "cos . . . " label has been manually improved.

11.4 Automatic vs. Manual Example

Fig. 9a has been created using strictly the automatics provided through `PSfragExport`, while fig. 9b achieves the same result by not relying on automatics at all but instead marking each textual element with a `PSfrag` command and exporting with `PSfragManualExport`. Both examples share the first lines of code that stretch over the full width.

```
f1[x_] := Sin[x];
f2[x_] := 3*((Cos[2 Sqrt[x]])^2)^(1/3);

SimpleLabel[tip : {_, _}, txt_, txtpos : {_, _}, align : {_, _}] :=  Sequence[
  Arrow[txtpos, tip, HeadScaling→Absolute, HeadLength→8, HeadCenter→{}0.6],
  Text[txt, txtpos, align]];

mygrid = Map[{#, {AbsoluteDashing[{0.1, 1}], GrayLevel[0.5]}} &, {Pi*{1/2, 1, 3/2}, {1, 2}}, {2}];
```

```
(* Automatic *)


tickmarks = {
  Pi/2*{0, 1, 2, 3, 4},
  Automatic,
  None,
  None};




textlabels = Graphics[{
  SimpleLabel[{Pi/2, f1[Pi/2]},
    "local maximum",
    {1, -0.5}, {0, 1}],
  SimpleLabel[{7/6Pi, f1[7/6Pi]},
    f1[x], {4.2, -0.3}, {-1, 0}],
  SimpleLabel[{4.2, f2[4.2]},
    f2[x],
    {3.5, 1.5}, {1, 0}]
  }];

rawplot = Plot[
  {f1[x], f2[x]}, {x, 0, 2 Pi},
  PlotStyle→{Hue[1.0], Hue[0.6]},
  Frame→True, FrameTicks→tickmarks,
  TextStyle→{FontFamily→"Times"}];

exampleplot = Show[rawplot, textlabels,
  GridLines→mygrid];

PSfragExport["ex_auto", exampleplot];

Export["ex_nopsfrag.eps", exampleplot];
```

```
(* Manual *)


mantickmarks = {
  {N[#], PSfrag[#, TeXPosition→"tc"]} &
    /@ (Pi/2*{0, 1, 2, 3, 4}),
  {N[#], PSfrag[#, TeXPosition→"cr"]} &
    /@ {-1, 0, 1, 2, 3},
  None, None};

mantexstr = "$3\left|\cos \sqrt{4x}\right|^\frac{2}{3}$";

mantextlabels= Graphics[{
  SimpleLabel[{Pi/2, f1[Pi/2]},
    PSfrag["local maximum", TeXPosition→"tc"],
    {1, -0.5}, {0, 1}],
  SimpleLabel[{7/6Pi, f1[7/6Pi]},
    PSfrag[f1[x], TeXPosition→"cl"], {4.2, -0.3}, {-1, 0}],
  SimpleLabel[{4.2, f2[4.2]},
    PSfrag[f2[x], TeXPosition→"cr", TeXCommand→mantexstr],
    {3.5, 1.5}, {1, 0}]
  }];

manrawplot = Plot[
  {f1[x], f2[x]}, {x, 0, 2 Pi},
  PlotStyle→{Hue[1.0], Hue[0.6]},
  Frame→True, FrameTicks→mantickmarks
  ];

Show[manrawplot, mantextlabels,
  GridLines→mangrid];

PSfragManualExport["ex_manual", %];
```

12 How to bug report

There is a number of things you can do/tell me to speed up the process of finding the error:

1. What is your operating system (Windows, Linux, Mac OS)?
2. What is your MATHEMATICA version (4.x, 5.x, 6.x)?
3. What are the steps that lead to the error? Please create a *minimal* example. Just copy all lines needed (and *only* those) to reproduce the error into a fresh notebook. Then think about if there are lines that can still be removed. Then quit your kernel and evaluate the notebook in one rush.
4. Send me an email (see first page of this manual).
5. Send me your credit card number.[5]

[5] Just kidding, of course I only accept cash.

A Program Summary

Manuscript Title:	MathPSfrag: Creating Publication-quality Labels in MATHEMATICA® Plots
Authors:	Johannes Große
Program Title:	MathPSfrag
Journal Reference:	
Catalogue identifier:	
Program available from:	http://wwwth.mppmu.mpg.de/members/jgrosse/mathpsfrag
Licensing provisions:	CPC non-profit use license
Programming language:	MATHEMATICA 6.0, 5.2, 5.0, 4.1
Computer:	Tested on x86 architecture
Operating systems:	Tested on Linux, Windows XP
RAM used during test run:	11 Mb
Keywords:	Encapsulated PostScript, MATHEMATICA, PSfrag, LaTeX
PACS:	01.30.Rr
Classification:	14 Graphics
External routines:	Standard MATHEMATICA packages
Nature of problem:	Insufficient typesetting quality of labels in graphics exported from MATHEMATICA
Solution method:	An automatic export function is provided that generates LaTeX substitution labels.
Requirements:	LaTeX, Ghostscript, PSfrag; recommended: `ps2eps`, `pdfcrop`, Perl
Restrictions:	The described method requires to some extent the use of Encapsulated PostScript (EPS) graphics though conversion to PDF is supported. Special MATHEMATICA characters that do not have a direct counterpart in LaTeX will not show satisfactory typesetting quality. For MATHEMATICA versions earlier than 5.1, the automatically created LaTeX code requires a compatibility LaTeX package, which is included in the package. In MATHEMATICA 4.1, one of the examples does not work in its current form due to a bug in MATHEMATICA's `Export` command. However, the same effect can be achieved using exclusively MathPSfrag's rotation mechanism.
Running time:	41s for all examples in this article

References

[1] J. Große, MathPSfrag: Publication quality labels with Mathematica. 1

[2] W. McKay, R. Moore, Convenient Labelling of Graphics, the WARMreader Way, TUGboat 20(3).
URL http://www.tug.org/TUGboat/Articles/tb20-3/tb64ross.pdf 3

[3] M. C. Grant, D. Carlisle, The PSfrag system, version 3.
URL http://www.ctan.org/tex-archive/macros/latex/contrib/psfrag/pfgguide.ps 3, 4, 6, 7

[4] S. Wolfram, The Mathematica book (4th edition), Cambridge University Press, New York, NY, USA, 1999. 3

[5] Wolfram Research, Inc., Mathematica 5.2 (2005).
URL http://www.wolfram.com/ 3

[6] WRI Support, tetex 1.0 with mathematica?
URL http://support.wolfram.com/mathematica/interface/export/tetexconfig.html 3, 26

[7] WRI Support, Mathematica fonts and ghostscript.
URL http://support.wolfram.com/mathematica/graphics/export/ghostscript.html 3, 26

[8] WRI Support, Mathematica fonts in eps files.
URL http://support.wolfram.com/mathematica/graphics/export/includefonts.html 3, 26

[9] J. Große, MathPSfrag.
URL http://wwwth.mppmu.mpg.de/members/jgrosse/mathpsfrag/ 3

[10] M. Caprio, Custom tick marks for linear, logarithmic, and general nonlinear axes (2005).
URL http://library.wolfram.com/infocenter/MathSource/5599/ 4, 5, 8, 29

[11] R. Niepraschk, H. Gäßlein, The pst-pdf package.
URL http://www.ctan.org/tex-archive/graphics/pstricks/contrib/pst-pdf/pst-pdf.pdf 4

[12] A. Chambert-Loir, R. CV, R. CV, pdftricks.
URL http://www.ctan.org/tex-archive/macros/latex/contrib/pdftricks/ 4

[13] T. Tantau, The BEAMER *class*.
URL http://latex-beamer.sourceforge.net/ 4

[14] H. Obeberdiek, pdfcrop.
URL http://www.ctan.org/tex-archive/support/pdfcrop/ 4

[15] Red Hat, Inc., Homepage of the GPL'ed cygwin version.
URL http://www.cygwin.com 6, 16

Index