

feynMF:
Drawing Feynman Diagrams
with \LaTeX{} and METAfont*

Thorsten Ohl†

Technische Hochschule Darmstadt
Schloßgartenstr. 9
D-64289 Darmstadt
Germany

December 30, 1997

Abstract

feynMF is a \LaTeX{} package for easy drawing of professional quality Feynman diagrams with METAfont (or META O T). FeynMF lays out most diagrams satisfactorily from the structure of the graph without any need for manual intervention. Nevertheless all the power of METAfont (or META O T) is available for obscure cases.

Copying

feynMF is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version. FeynMF is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

*This is feynmf.dtx, version v1.08, revision 1.30, date 1996/12/02.
†e-mail: Thorsten.Ohl@Physik.TH-Darmstadt.de
Contents

1 Introduction .......................... 4
  1.1 Purpose and scope ................................. 4
  1.2 Relation to similar packages ......................... 5
  1.3 Historical note ................................ 6
  1.4 Architecture .................................. 6
  1.5 Conclusion .................................. 6

2 Usage ................................... 8
  2.1 \LaTeX{} package and environments .................... 8
  2.2 Auxiliary files ................................ 9
  2.3 Running METAFONT .......................... 10
  2.4 The FeynMF perl script ......................... 12
  2.5 Vertex mode .................................. 13
    2.5.1 External vertices .......................... 14
    2.5.2 Arcs and internal vertices ..................... 14
    2.5.3 Polygons ................................ 18
    2.5.4 Color ................................ 19
    2.5.5 Examples ................................ 19
    2.5.6 Labels ................................ 23
    2.5.7 Manipulating the layout ....................... 24
    2.5.8 Skeletons ............................... 25
    2.5.9 Pulling strings ........................... 27
  2.6 Miscellaneous commands .......................... 29
    2.6.1 Graphs in graphs .......................... 29
    2.6.2 Reusing diagrams .......................... 30
    2.6.3 Grouping ................................ 30
    2.6.4 Changing parameters ........................ 31
    2.6.5 Shrinking .............................. 31
    2.6.6 Debugging .............................. 31
    2.6.7 Multiple vertices and arcs .................... 31
  2.7 Immediate mode ............................ 32
    2.7.1 Arcs ................................ 33
    2.7.2 Vertices ................................ 33
    2.7.3 Declarations ............................. 33
    2.7.4 Assignments ............................. 34
    2.7.5 Examples ................................ 34
  2.8 Raw METAFONT .............................. 35
    2.8.1 Extending FeynMF ........................ 36
  2.9 Common traps, trouble shooting and frequently asked questions (FAQs) ......... 39
    2.9.1 ! Value is too large ........................ 39
    2.9.2 Diagrams in the document are never updated .... 40
    2.9.3 Diagrams show up in the wrong spot .......... 40
    2.9.4 Spurious labels show up ..................... 40
  2.10 Known bugs ................................ 41
    2.10.1 Chaotic manual .......................... 41
    2.10.2 Delayed error messages ..................... 41
    2.10.3 Multiple tadpoles ........................ 41

2

2.10.4 Hard limits ........................................ 41
1 Introduction

1.1 Purpose and scope

In recent years, \TeX\[^{1}\] and \LaTeX\[^{2}\] have revolutionized the way we share information in theoretical physics (and other areas). Not only does \TeX provide typographical capabilities, which transcend those of commercial “wordprocessors” substantially, \TeX documents are also completely portable. Since implementations are available on essentially all computers in use in the community, documents can be shared without the usual restrictions of proprietary data formats. This has enabled us to collaborate on papers with colleagues on the other side of the globe, to replace the mailing of hard copy preprints by electronic transmission and to submit these papers electronically to the publisher.

This portability comes with a price, though. \TeX (and \LaTeX) do not address the issue of graphical information, apart from the rudimentary (but very useful) capabilities of the \LaTeX picture environment and similar packages \[^{3}\]. As an de facto standard for the inclusion of more complex graphics has emerged the inclusion of PostScript[^4] files. The complete document can then be printed on any PostScript device.

Still there are areas, where complementary approaches seem worth pursuing. In particular this is the case, if the graphical information is highly formalized, like the case at hand. Feynman diagrams are specified by their topology and the type of particles connecting the vertices. Thus a given diagram can be reproduced from a very concise specification, if the software is able to choose a reasonable layout (semi-)automatically.

\METAFONT[^5] and METRA \O T \[^{5}\] appear to be the perfect tool for such a purpose, since

1. \METAFONT is part of any (reasonable) \TeX installation, thus available to all potential users,

2. both have very powerful graphics primitivs, which allow high quality output, and

3. both have builtin linear algebra, which allows us to choose a layout automatically.

Still, providing at least the basic interface in \LaTeX macros seems appropriate for boosting the acceptance among the less technically oriented parts of the audience. Thus \feynMF[^6] \[^{6}\] was conceived.

\feynMF is unique among packages for drawing Feynman diagrams in combining the following features:

- Simplicity and conciseness for common diagrams. E.g. the scattering diagram in figure 1 can be specified completely in five lines of \LaTeX:

---

[^1]: \TeX is a trademark of the American Mathematical Society.
[^2]: \LaTeX might be a trademark of Addison Wesley Publishing Company.
[^3]: PostScript is a trademark of Adobe Systems Inc.
[^4]: \METAFONT is a trademark of Addison Wesley Publishing Company.
[^5]: John Hobby’s METRA \O T is a modified version of \METAFONT which generates (encapsulated) PostScript output. METRA \O T can be build trivially on top of the web2c version of \TeX and \METAFONT for UNIX. Ports to other systems should be simple.
[^6]: \feynMF is not anybody’s trademark.
Figure 1: Simple scattering diagram.

\begin{fmfgraph*}(40,30) \fmfpen{thick}
\fmfleft{i1,i2} \fmfright{o1,o2}
\fmf{fermion}{i1,v1,o1} \fmf{fermion}{i2,v2,o2}
\fmf{photon,label=$q$}{v1,v2} \fmfdot{v1,v2}
\end{fmfgraph*}

- Expressiveness for complicated diagrams (see the examples below).
- Extensibility (e.g. see section 2.8.1).
- Arbitrary \TeX-labels. This point is more important than it may seem at first glance because most graphical layout systems lack the power to produce complicated mathematical expressions. Having matching fonts in text, equations and diagrams is also an important esthetical feature.

1.2 Relation to similar packages

Before we start, a couple of words about some complementary packages on the market are in order:

- Michael Levine’s \texttt{feynman} package [8] is implemented on top of the standard \LaTeX picture environment [2]. This makes it completely portable (no need for a correct \textsc{Metafont} installation), but it requires manual layout and the graphics output is (though very useful) less than perfect.

- Jos Vermaseren’s \texttt{axodraw} package [9] uses \specials to access PostScript primitives for drawing diagrams. This approach is inherently not portable (the ubiquity of PostScript printers makes this a minor point, though) but as flexible as the present one. Nevertheless, it still requires manual layout for all diagrams.

- Last, but not least, I have to mention Thomas Leathrum’s \texttt{mfic} [10], which provided the inspiration for moving \texttt{feynMF}’s user interface from \textsc{Metafont} to \LaTeX. It might not have been unreasonable to design and implement \texttt{feynMF} as a package on top of \texttt{mfic}, but closer inspection shows that \texttt{feynMF} and \texttt{mfic} are fairly orthogonal. \texttt{mfic} is most useful for handling simple graphical building blocks in formally unconstrained contexts. \texttt{feynMF} on the other hand excels in the formal context of Feynman diagrams (or any other labeled graphs for that matter), which can be drawn automatically from a \textit{specification}.
1.3 Historical note

Parts of this code have a rather long history\(^7\). Some of the drawing primitives started in 1989 as \texttt{feynman.mf}, a library of \textsc{metafont} macros for drawing Feynman diagrams in my thesis. The layout had to be specified completely by hand, which required a long edit-process-preview cycle and made \texttt{feynman.mf} awkward to use. Nevertheless, it suited my and other's needs and survived for five years without major modifications. Early in 1994, I became aware of Thomas Leathrum's \texttt{mfpic}[10]. This inspired me to shift the user interface from \textsc{metafont} to \textsc{latex}, because this allows a smoother blending of the \textsc{latex} picture environment with \texttt{feynMF} for the purpose of labeling the graphs. While doing this and after having been taught by Tim Stelzer's and Bill Long's \textsc{madgraph}[11] that simply minimizing the length of the graph gives much better results than I had anticipated, I added the graph manipulation and layout code.

1.4 Architecture

Even though there has never been a proper design phase in the development of \texttt{feynMF}, a certain structure has emerged, which is depicted in figure 2. A user who is aware of this architecture should be able to use \texttt{feynMF} more effectively. The most crucial aspect of the architecture is the existence of two distinct modes with different fundamental datatypes:

- \textit{vertex mode}, which deals with graphs as structures consisting of vertices and arcs and (almost) never deals with their physical locations.

- \textit{immediate mode}, which deals with \textsc{metafont} paths and pairs (i.e. coordinates) and allows complete control over the physical locations.

It is of course possible to mix these modes in advanced applications. Commands are provided to translate vertices and arcs to \texttt{pairs} and \texttt{paths} and vice versa. Novice users with little experience in \textsc{metafont} programming should start with vertex mode to get their job done. Later, immediate mode can be used to create more and more complex diagrams. It is possible to create most diagrams that can actually be calculated in vertex mode. Immediate mode is most useful for extending line \texttt{feynMF} and for drawing diagrams with fancy decorations.

A word on portability: \texttt{feynMF} is implemented as a \textsc{latex} package. But it should be straightforward to adapt it to other \textsc{latex} macro packages because \textsc{latex} macros have been used for convenience only and can easily be replaced or provided in a compatibility package. The \textsc{latex} specific \texttt{environment} construct can also easily be replaced by the equivalent construct in another macro package.

1.5 Conclusion

It goes without saying that \texttt{feynMF} is not perfect. There might be cases where using a graphical drawing tool with a mouse can give more pleasing results in less time. But in most cases, \texttt{feynMF} will give satisfactory results without any fine tuning. These will be reproducible and independent from the computer it is running on.

\(^7\text{Which is a partial explanation, if not excuse, for its slightly incoherent structure.}\)
Figure 2: Architecture of \texttt{feynMF}: the two modes of \texttt{feynMF} (immediate mode and vertex mode) both interact with \LaTeX{} and METAFONT (or META OT, respectively), but operate on different data types.

Early user responses have been very encouraging. There seems to be a relatively steep learning curve for those \LaTeX{} users that have to find out how to run METAFONT on their systems. But once this purely technical obstacle has been surmounted, users have been enthusiastic as well about the quality of the generated graphs as about the ease of use of \texttt{feynMF}.
2 Usage

In addition to this manual, there exists also a concise description of \texttt{feynMF} in a journal article \cite{6}, as well as a three part tutorial \cite{7}.

2.1 \LaTeX\ package and environments

Instructing \LaTeX\ to use \texttt{feynMF} is as simple as

\begin{verbatim}
\usepackage{feynmf}
\end{verbatim}

If you have \texttt{METAFONT}, then you can use it alternatively by placing

\begin{verbatim}
\usepackage{feynmp}
\end{verbatim}

in your \LaTeX\ source.\footnote{As given, this applies to \LaTeX. But the installation file \texttt{feynmf209.ins} allows to generate special versions \texttt{feynmf209.sty} and \texttt{feynmp209.sty} which are compatible with the obsolete \LaTeX\ version 2.09. These files are to be used as \texttt{documentstyle} options

\begin{verbatim}
\documentstyle[,...,feynmf209,...]{...}
\end{verbatim}

or

\begin{verbatim}
\documentstyle[,...,feynmp209,...]{...}
\end{verbatim}

If you cannot use \texttt{epsf.sty} for including PostScript files, you can either hack \texttt{feynmp209.sty} or upgrade to \LaTeX\2e. Please keep in mind that \texttt{feynMF} has been developed for \LaTeX\2e and the \LaTeX\ 2.09 compatibility version will always be a retrofitted hack. I will accept bug reports for the 2.09 version, but I urge everybody to move to \LaTeX\2e, which is the one and only supported \LaTeX\ right now.\footnote{\texttt{feynMF} understands an option \texttt{pre-1.03}, that is of interest for veteran users:}

\begin{verbatim}
\usepackage[pre-1.03]{feynmf}
\end{verbatim}

or

\begin{verbatim}
\usepackage[pre-1.03]{feynmp}
\end{verbatim}

The purpose of this option is to enable processing of old input files (pre v1.02) that use \texttt{\noexpand} in labels. Since the method for processing these files can clash (in rare cases) with \LaTeX\2e’s font loading procedure, it has been disabled by default.}

\begin{verbatim}
\begin{fmffile}{(METAFONT-file)}
...
\end{fmffile}
\end{verbatim}

All descriptions that should go into one \texttt{METAFONT} file are placed inside a \texttt{fmffile} environment which takes the name of the \texttt{METAFONT} file as an argument:
Upto 255 graphs can be placed into one \texttt{METAFONT} file. Currently \texttt{feynMF} does \textit{not} check that the 255 graph limit per file is not overrun.\footnote{There is also a very primitive, but (unfortunately) popular operating system, which restricts filenames to eight characters with a three character extension. On this system, only 99 graphs can be placed into one \texttt{METAFONT} file because auxiliary files will not be unambiguous, if more than two digits are used.} Note that the filename for the \texttt{METAFONT} file given in the argument of the \texttt{fmffile} environment \textit{must not} be identical to the \LaTeX{} source file name, because the \texttt{METAFONT} \texttt{.log} would be overwritten and \LaTeX{} can no longer access the information in this \texttt{.log} file. It should be obvious that anyumber of diagrams can be generated by using more than one \texttt{fmffile} environment with different filenames.

\texttt{fmffile} The \texttt{fmffile} environment contains the description of a single Feynman diagram which will be placed at the location of the environment. Arguments are the width and the height of the diagram, in units of \texttt{\unitlength}:

\begin{verbatim}
\begin{fmffile}(\textit{width},\textit{height})
...
\end{fmffile}
\end{verbatim}

This environment does \textit{not} support labels, use \texttt{fmffile*} if your diagrams contains labels.

\texttt{fmffile*} Same as \texttt{fmffile}, but enclosed in a \texttt{picture} environment of the same size and supporting \LaTeX{} labels.

\begin{verbatim}
\begin{fmffile*}(\textit{width},\textit{height})
...
\end{fmffile*}
\end{verbatim}

\texttt{fmfframe} Allows to allocate additional space around a \texttt{fmffile*}, since the labels (or the diagram itself) might overshoot:

\begin{verbatim}
\begin{fmfframe}(\textit{left},\textit{top})(\textit{right},\textit{bottom})\{\textit{box}\}
\end{fmfframe}
\end{verbatim}

puts an invisible frame of the given dimensions (measured in \texttt{\unitlength}) around \texttt{\{box\}}.

\section{2.2 Auxiliary files}

\texttt{feynMF} needs to share information between \texttt{METAFONT} and \LaTeX{}. For this task several auxiliary files are needed. The flow of information depicted in figures 3 and 4 looks much more complicated than it is. The important feature is that there are two sets of files which can be used to distribute a document:

1. If the recipient has a working \texttt{METAFONT} installation (which shouldn’t be a problem, except for some impoverished commercial implementations), the document can be typeset from the \LaTeX{} source \textit{alone}, by running \LaTeX{}, \texttt{METAFONT} and \LaTeX{} again (the latter step might have to be repeated to get cross references right).

2. Another possibility (which doesn’t require \texttt{METAFONT} on the recipient’s side), is to distribute the \LaTeX{} source, the \texttt{tfm} and \texttt{gf} files (or \texttt{pk} files respectively) along with the label files with extension \texttt{tn} (where \texttt{n} as an
2.3 Running METAFONT

Processing your document with \LaTeX will generate one or more METAFONT files, which you will have to process with METAFONT. On UNIX\textsuperscript{11} systems, METAFONT is invoked as

\begin{verbatim}
  mf '\mode:=(METAFONT-mode); input (METAFONT-file)'
\end{verbatim}

from the shell. Here \texttt{(METAFONT-file)} is to be replaced by the name of the input file (which is determined by the argument to the \texttt{fmffile} environment, see section 2.1) and \texttt{(METAFONT-mode)} is the proper METAFONT-mode for your particular printer. Please consult your local guide or local wizards about how to run METAFONT on other systems.

\textsuperscript{11}UNIX was a trademark of UNIX Systems Laboratory, but is rumored to have been donated to X/Open.
Figure 4: Interdependency of files in a \texttt{feynMF} application using META \O T: The arrows show which files are updated in the two \LaTeX passes, the META-\O T pass and the final dvi translation step.

Note that (\texttt{METAFONT-mode}) must be specified, otherwise \texttt{METAFONT} will fail or the resulting font will not be usable.\textsuperscript{12} You can look up the correct \texttt{METAFONT} mode in the file \texttt{modes.mf} that comes with the \texttt{METAFONT} distribution. Among the more common laser printers are \texttt{laserjet} for HP Laserjets at 300dpi, \texttt{ljfour} for HP Laserjets at 600dpi, \texttt{nexthi} for NeXT laser printers at 400dpi, etc.

A non-trivial part can be instructing \LaTeX and your favorite dvi-driver how to find the generated \texttt{tfm} and \texttt{gf} (resp. \texttt{pk}) files. This is highly system dependent and can be trivial (as in the standard UNIX \TeX installations, where no further action is required) or almost impossible without system privileges (as under MVS). Please consult your local guide or wizards on this point. See also section 2.9.2 for common problems with dvi drivers.

Some recent \TeX implementations (e.g. \texttt{web2c} with \texttt{kpathsea} version 2.6 or later) are able to generate \texttt{tfm} files on the fly. Using such implementations, running \TeX twice should suffice and \texttt{METAFONT} will be invoked automagically in the background. Note however, that the automagically invoked tools might also

\textsuperscript{12}See section 2.9.1 for the typical error message and for additional information on printer modes.
install the “fonts” corresponding to the Feynman diagrams in a system directory, where they don’t belong. Adding the following lines to the `maketex.site` script will prevent this mishap in the `tex` distribution for UNIX (which is derived from web2c):

```bash
if [ -r $KPSE_DOT/$NAME.mf ]; then
    MT_PKDESTDIR=$KPSE_DOT
    MT_TFMDESTDIR=$KPSE_DOT
    MT_NAMEPART=
fi
```

The automagic tools will also *not* notice when a diagram has changed. These problems suggest that it is a generally a good idea to invoke `Metafont` explicitly, instead of relying on the automagic tools.

Running `Meta O T` is usually trivial, because not printer specific mode is needed:

```
mp (Meta O T-file)
```

### 2.4 The `feynmf` perl script

UNIX users will be able to take advantage of the `feynmf` perl script, that automates the invocation of `LaTeX` and `Meta O T`. In particular it tries to guess the correct `Metafont`-mode and magnification. The latter is often different from 1 in slide classes. Here is the man page of `feynmf`:

**NAME**

`feynmf` — Process `LaTeX` files using `FeynMF`

**SYNOPSIS**

```
feynmf [-hvqncfT] [-t tfm [-t tfm ...]] [-m mode] file [file ...]
        [--notfm] [--tfm tfm [--tfm tfm ...]] [--mode mode] file [file ...]
```

**DESCRIPTION**

The most complicated part of using the `FeynMF` style appears to be the proper invocation of `Metafont`. The `feynmf` script provides a convenient front end and will automagically invoke `Metafont` with the proper mode and magnification. It will also avoid cluttering system font directories and offers an option to clean them.

**OPTIONS**

- `-h`, `--help`
  
  Print a short help text.

- `-v`, `--version`
  
  Print the version of `feynmf`. 

12
-q, --quiet
Don’t echo the commands being executed.

-n, --noexec
Don’t execute LaTeX or Metafont.

-c, --clean
Offer to delete font files that have accidentally been placed in a system directory by the MakeTextTFM and MakeTeXPK scripts (these scripts are run by tex (and latex) in the background). This option has only been tested with recent versions of UNIX TeX.

-f, --force
Don’t ask any questions.

-T, --notfm
Don’t try to prepare fake .tfm files for the first run.

-t, --tfm tfm
Don’t try guess the names of the .tfm files to fake for the first run and use the given name(s) instead. This option can be useful if our incomplete parsing of the LaTeX input files fails.

-m mode, --mode mode
Select the METAFONT mode mode. The default is guessed or localfont if the guess fails.

file
Main LaTeX input files.

file ...
Other LaTeX input files that are included by the main file.

AUTHOR
Thorsten Ohl <Thorsten.Ohl@Physik.TH-Darmstadt.de>

BUGS
The preparation of .tfm files is not foolproof yet, because we can parse TeX files only superficially. This script has only been tested for recent teTeX distributions of UNIX TeX, though it will probably work with other versions of UNIX TeX. The author will be grateful for portability suggestions, even concerning Borg operating systems, for the benefit of those users that are forced to live with DOS or Windows.

2.5 Vertex mode
These basic features of feynMF are (or rather “should be”) available through the \LaTeX interface. No knowledge of METAFONT is necessary.
2.5.1 External vertices

Positioning of external vertices has to done explicitly. The technical reason is that they would otherwise collapse with their neighbors, but practical reasons also suggest to give the user full control here. \fml\{v1\}, \ldots\} places the vertices in the comma separated list \{v1\},... equidistantly on a smooth path on the left side of the diagram. \fmr\{v1\}, \ldots\} does the same thing on the right. Similarly \fmb\{v1\}, \ldots\} and \fmt\{v1\}, \ldots\} places its arguments on smooth path surrounding the diagram. Per default, the galleries on which we place the external vertices are curved as in figure 5, but straight galleries are also available. The macros \fmc\ and \fms\ switch between these alternatives.

\fmln\ The macro \fmln\ is similar to \fml, but \fmln\{v\}, \ldots\} places the vertices \{v1\}, \ldots\} and \fmsn\{v\}, \ldots\} and \fmbn\{v\}, \ldots\} and \fmtn\{v\}, \ldots\}.

2.5.2 Arcs and internal vertices

\fmm This is the the most frequently used macro in feynMF applications.

\fmm\{\langle style\rangle,\langle opt\rangle=\langle val\rangle,\ldots\}\{\langle v1\rangle,\langle v2\rangle,\ldots\}\}

connects the vertices \{v1,v2,\ldots\} with a line of style \langle style\rangle, using a set of options \langle opt\rangle with (optional) value \langle val\rangle. If a vertex is not known yet, it is added to the diagram. Note that the actual drawing is not done immediately, because the positions can only be calculated when all vertices are known. The currently available styles are collected in table 1. Most names should be self explanatory and are not discussed further. The \texttt{dashes}, \texttt{dots}, \texttt{phantom} and \texttt{plain} styles can optionally be decorated with an arrow as shown above. All styles, including \texttt{curly}, \texttt{wiggly} and \texttt{zigzag}, can be doubled. But arrows are not available for the latter three, because esthetically pleasing results can not be expected. The \texttt{phantom} style is special, because it only enters the vertices and does not cause a line to be drawn. This is extremely useful for advanced layout features, as explained below. If you need a line styles that is not listed in table 1, see section 2.8.1 for how to define your own line styles.
<table>
<thead>
<tr>
<th>Name</th>
<th>Example</th>
<th>Parameters</th>
<th>Aliases</th>
</tr>
</thead>
<tbody>
<tr>
<td>curly</td>
<td><img src="example" alt="curly" /></td>
<td>curly_len</td>
<td>gluon</td>
</tr>
<tr>
<td>dbl_curly</td>
<td><img src="example" alt="dbl_curly" /></td>
<td>curly_len</td>
<td></td>
</tr>
<tr>
<td>dashes</td>
<td><img src="example" alt="dashes" /></td>
<td>dash_len</td>
<td></td>
</tr>
<tr>
<td>dashes_arrow</td>
<td><img src="example" alt="dashes_arrow" /></td>
<td>dash_len</td>
<td>scalar</td>
</tr>
<tr>
<td>dbl_dashes</td>
<td><img src="example" alt="dbl_dashes" /></td>
<td>dash_len</td>
<td></td>
</tr>
<tr>
<td>dbl_dashes_arrow</td>
<td><img src="example" alt="dbl_dashes_arrow" /></td>
<td>dash_len</td>
<td></td>
</tr>
<tr>
<td>dots</td>
<td><img src="example" alt="dots" /></td>
<td>dot_len</td>
<td></td>
</tr>
<tr>
<td>dots_arrow</td>
<td><img src="example" alt="dots_arrow" /></td>
<td>dot_len</td>
<td>ghost</td>
</tr>
<tr>
<td>dbl_dots</td>
<td><img src="example" alt="dbl_dots" /></td>
<td>dot_len</td>
<td></td>
</tr>
<tr>
<td>dbl_dots_arrow</td>
<td><img src="example" alt="dbl_dots_arrow" /></td>
<td>dot_len</td>
<td></td>
</tr>
<tr>
<td>phantom</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>phantom_arrow</td>
<td><img src="example" alt="phantom Arrow" /></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plain</td>
<td></td>
<td></td>
<td>vanilla</td>
</tr>
<tr>
<td>plain_arrow</td>
<td><img src="example" alt="plain Arrow" /></td>
<td></td>
<td>fermion, electron, quark</td>
</tr>
<tr>
<td>dbl_plain</td>
<td></td>
<td></td>
<td>double</td>
</tr>
<tr>
<td>dbl_plain_arrow</td>
<td><img src="example" alt="dbl_plain Arrow" /></td>
<td></td>
<td>double_arrow, heavy</td>
</tr>
<tr>
<td>wiggly</td>
<td><img src="example" alt="wiggly" /></td>
<td>wiggly_len</td>
<td>boson, photon</td>
</tr>
<tr>
<td>dbl_wiggly</td>
<td><img src="example" alt="dbl_wiggly" /></td>
<td>wiggly_len</td>
<td></td>
</tr>
<tr>
<td>zigzag</td>
<td><img src="example" alt="zigzag" /></td>
<td>zigzag_width</td>
<td></td>
</tr>
<tr>
<td>dbl_zigzag</td>
<td><img src="example" alt="dbl_zigzag" /></td>
<td></td>
<td>zigzag_len</td>
</tr>
</tbody>
</table>

Table 1: Available line styles

The supported options are collected in table 2. Note that each of the dot separated components of the options can be abbreviated. For example, l.d is equivalent to label.dist. The result of ambiguous matches is however undefined.

One particularly useful further option would be smooth, allowing for several lines joined smoothly. Early experimentation has shown however, that the results are not always what one expects and that there is a lot of room for abuse.
<table>
<thead>
<tr>
<th>Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>tension</td>
<td>draw a tighter (&gt; 1) or more loose (&lt; 1) arc</td>
</tr>
<tr>
<td>left</td>
<td>draw on a halfcircle on the left</td>
</tr>
<tr>
<td>right</td>
<td>draw on a halfcircle on the right</td>
</tr>
<tr>
<td>straight</td>
<td>draw on a straight line (default)</td>
</tr>
<tr>
<td>label</td>
<td>\LaTeX text for labeling the arc</td>
</tr>
<tr>
<td>label.side</td>
<td>force placement of the label on the left or right</td>
</tr>
<tr>
<td>label.dist</td>
<td>place label at a distance dist</td>
</tr>
<tr>
<td>label.dist</td>
<td>relative position of the label (not implemented yet!)</td>
</tr>
<tr>
<td>tag</td>
<td>optional tag for disambiguating arcs</td>
</tr>
<tr>
<td>width</td>
<td>width of the line</td>
</tr>
<tr>
<td>rubout</td>
<td>scale factor for crossing out lines (doesn’t work properly for doubled lines yet)</td>
</tr>
<tr>
<td>foreground</td>
<td>foreground color (META O T only!)</td>
</tr>
<tr>
<td>background</td>
<td>background color for doubled lines (META O T only!)</td>
</tr>
</tbody>
</table>

Table 2: Available line options

Note that because the options are separated by single commas, commas inside arguments to options (\texttt{label} comes to mind) have to be doubled (similar to quotes in Fortran).

Arcs that return to their origin are allowed (I will refer to them as \texttt{tadpoles}), but some options have slightly different semantics. \texttt{tension} is here a inverse scale factor for the tadpole, whose default size is 2/3 of the average distance the neighboring vertices. If \texttt{left} or \texttt{right} are specified, they give the direction (in degrees) of the preferred gap into which the tadpole is placed. By default, the largest gap is chosen for all tadpoles at a given vertex, which will therefore overlap. This is neither a bug nor a feature, but a limitation.

\begin{verbatim}
\fmfn The macro \texttt{\fmfn} is similar to \texttt{\fmf}, but
\fmfn{\langle style\rangle,\langle opt\rangle=\langle val\rangle,\ldots}\langle (v)\rangle\langle (n)\rangle
connects the vertices \{v[1]\ldots v[n]\}.
\end{verbatim}

\begin{verbatim}
\fmfcyclen The macro \texttt{\fmfcyclen}\langle (v)\rangle\langle (n)\rangle cyclically connects the vertices
\{v[1]\ldots v[n]\}. \texttt{\fmfcyclen} operates in reverse order.
\end{verbatim}

\begin{verbatim}
\fmfpen Pick up a pen of the specified size. \texttt{\fmfpen\langle weight\rangle} is used for changing the
weight (i.e. thickness) of the lines. Predefined sizes are \texttt{thin} and \texttt{thick}.
\end{verbatim}

\begin{verbatim}
\fmfv Declare vertices with options:
\fmfv{\langle opt\rangle=\langle val\rangle,\langle opt\rangle=\langle val\rangle,\ldots}\{v[1],\ldots\}
This is used for adding labels to a vertex and for specifying other decoration.
Supported options are collected in table 3. Here the same abbreviation mecha-
\end{verbatim}

\footnote{Note that, as of version 1.03, it is no longer necessary to escape \LaTeX control sequences in arguments. Old files will continue to work, because \texttt{noexpand} is temporarily disabled.}
Table 3: Available vertex options

<table>
<thead>
<tr>
<th>Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>label</td>
<td>(\LaTeX) text for labeling the vertex</td>
</tr>
<tr>
<td>label.angle</td>
<td>force placement of the label at the given angle from the vertex</td>
</tr>
<tr>
<td>label.dist</td>
<td>place label at a distance (\text{dist})</td>
</tr>
<tr>
<td>decoration.shape</td>
<td>shape of decoration</td>
</tr>
<tr>
<td>decoration.size</td>
<td>size of decoration</td>
</tr>
<tr>
<td>decoration.filled</td>
<td>fill, shade or hatch decoration</td>
</tr>
<tr>
<td>decoration.angle</td>
<td>rotate decoration</td>
</tr>
<tr>
<td>foreground</td>
<td>foreground color (META (\text{O T}) only!)</td>
</tr>
<tr>
<td>background</td>
<td>background color (META (\text{O T}) only!)</td>
</tr>
</tbody>
</table>

nim as above is in effect. The available shapes are listed in various filling styles in table 4. The tilings \texttt{gray10}, \texttt{gray25}, \texttt{gray75} and \texttt{gray90} are available in addition to \texttt{gray50}. Customized tilings can be created with the \texttt{METAFONT} function \texttt{tile_from_string}. It should be noted however, that tilings are gobbling up memory at high speed and should be used with discretion. The halftones can be accessed by giving any number from 2 to 99, which will denote the percentage of saturation (30\% and 70\% here). Again, commata inside arguments to options have to be doubled.

\texttt{\texttt{\textbackslash fmfblob}\ Draw a blob of the specified diameter at the vertices. Incidentally,}
\begin{verbatim}
\texttt{\textbackslash fmfblob\{(diameter)\}{\{v1\}[...]}\}
\end{verbatim}
is equivalent to
\begin{verbatim}
\texttt{\textbackslash fmfv\{decor.shape=circle,decor.filled=shaded,}
\texttt{decor.size=\{diameter\}\}{\{v1\}[...]}\}
\end{verbatim}

\texttt{\texttt{\textbackslash fmfdot}\ Draw a dot at the vertices given as arguments.}
\begin{verbatim}
\texttt{\textbackslash fmfdot\{\{v1\}[...]}\}
\end{verbatim}
is equivalent to
\begin{verbatim}
\texttt{\textbackslash fmfv\{decor.shape=circle,decor.filled=full,}
\texttt{decor.size=2thick\}{\{v1\}[...]}\}
\end{verbatim}

\texttt{\texttt{\textbackslash fmfvn}\ The macro \texttt{\textbackslash fmfvn} is similar to \texttt{\textbackslash fmfv}, but}
\begin{verbatim}
\texttt{\textbackslash fmfvn\{(opt)=[\{val\}][,\{opt\}=[\{val\}],...]\}{\{v\}\{n\}}\}
\end{verbatim}
places the vertices \{\texttt{v[1]}\}...\{\texttt{v[n]}\}.

\texttt{\texttt{\textbackslash fmfblobn}\ The macros \texttt{\textbackslash fmfdotn} and \texttt{\textbackslash fmfblobn} are similar to the \texttt{\textbackslash fmfdot} and \texttt{\textbackslash fmfblob}, but \texttt{\textbackslash fmfdotn}\{(\{v\})(\{n\})\} places the vertices \{\texttt{v[1]}\}...\{\texttt{v[n]}\}. Analogously for \texttt{\textbackslash fmfblobn}.

---

\footnote{If the variable \texttt{feynmwuiz} is \texttt{true} (e.g. after calling the \texttt{\textbackslash fmfwiz} macro), it is also possible to specify any \texttt{METAFONT} expression that evaluates to a \texttt{path}. Naturally, this has to used with great care, because strange errors can be triggered by typos!}

\footnote{\texttt{META \(\text{O T}\)} will give true halftones (if your printer supports them), while \texttt{METAFONT} tries to mimic them. The dithering algorithm of the latter will be improved in the future.}

\footnote{The old numeric arguments in the range \(-1\ldots1\) continue to work, but are considered obsolete.}
2.5.3 Polygons

Complex vertices with arcs attached at the corners can be constructed with polygons, which share some characteristics with arcs and vertices.

The macro `\fmfpoly{\langle style\rangle,\langle opt\rangle[=\langle val\rangle],\ldots}\{\langle v_1\rangle,\langle v_2\rangle[\ldots]\}` places the vertices $\langle v_1\rangle$, $\langle v_2\rangle$, $\ldots$ on the corners of a regular polygon. The orientation of the polygon is fixed to be mathematically positive (i.e. counter clockwise). Note that this can have strange results if the orientation is opposite to the orientation of the vertices the corners are connected to. The available options are collected in table 5. The effects of some options are depicted in table 6.

Note that it is technically impossible to fix the size of a polygon. The size can be controlled by specifying a tension for the edges, which acts like the tension of normal arcs.

The macro `\fmfpolyn\{\langle style\rangle\}\{\langle v\rangle\}\{\langle n\rangle\}` is similar to `\fmfpoly` but connects the vertices $\langle v[1]\rangle\ldots\langle v[n]\rangle$. `\fmfrpolyn` operates in reverse order.

<table>
<thead>
<tr>
<th>filled=</th>
<th>full</th>
<th>empty</th>
<th>shaded</th>
<th>hatched</th>
<th>gray50</th>
<th>30</th>
<th>70</th>
</tr>
</thead>
<tbody>
<tr>
<td>circle</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>square</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>triangle</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
</tr>
<tr>
<td>diamond</td>
<td>◆</td>
<td>◆</td>
<td>◆</td>
<td>◆</td>
<td>◆</td>
<td>◆</td>
<td>◆</td>
</tr>
<tr>
<td>pentagon</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>hexagon</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>triagram</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
<td>△</td>
</tr>
<tr>
<td>tetragram</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
</tr>
<tr>
<td>pentagram</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
</tr>
<tr>
<td>hexagram</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>triacross</th>
<th>cross</th>
<th>pentacross</th>
<th>hexacross</th>
</tr>
</thead>
<tbody>
<tr>
<td>♪</td>
<td>×</td>
<td>Ⱡ</td>
<td>Ⱡ</td>
</tr>
</tbody>
</table>

Table 4: Available vertex shapes and fill styles.
<table>
<thead>
<tr>
<th>Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>filled</td>
<td>fill, shade or hatch interior</td>
</tr>
<tr>
<td>phantom</td>
<td>don’t draw anything</td>
</tr>
<tr>
<td>empty</td>
<td>draw outline</td>
</tr>
<tr>
<td>shade</td>
<td>shade interior</td>
</tr>
<tr>
<td>hatched</td>
<td>hatch interior</td>
</tr>
<tr>
<td>full</td>
<td>fill interior</td>
</tr>
<tr>
<td>pull</td>
<td>pull edges in (&lt; 0) or out (&gt; 0)</td>
</tr>
<tr>
<td>tension</td>
<td>tension of the edges</td>
</tr>
<tr>
<td>smooth</td>
<td>draw smooth corners</td>
</tr>
<tr>
<td>label</td>
<td>(\text{\LaTeX}) text for labeling the polygon</td>
</tr>
<tr>
<td>label.angle</td>
<td>force placement of the label at the given angle from the vertex</td>
</tr>
<tr>
<td>label.dist</td>
<td>place label at a distance dist</td>
</tr>
<tr>
<td>foreground</td>
<td>foreground color (META O T only!)</td>
</tr>
<tr>
<td>background</td>
<td>background color (META O T only!)</td>
</tr>
</tbody>
</table>

Table 5: Available polygon options

2.5.4 Color

If META O T is used for rendering the diagrams, vertices and arcs can be colored. The corresponding options give a warning message under METAFONT and are otherwise ignored.

Two colors are available for vertices and arcs: `foreground` and `background`. Both can either be specified as a linear combination of the predefined colors `white`, `black`, `red`, `green` and `blue` or as RGB triples \((\text{red}), (\text{green}), (\text{blue})\). Note that as always commata have to be doubled in option arguments. Therefore both `foreground=(1,,0,,1)` and `foreground=red+blue` are valid options setting the foreground color to magenta. For arcs, the background color is used only for the interior of doubled lines.

While the color feature is rarely used in papers or books, it can be very useful for jazzing up your transparencies. See page 27 for a not very serious example.

2.5.5 Examples

After the main features of the vertex mode have been introduced, it is time for a couple of illustrative examples that are taken from *The Real World*.

As a first example, consider drawing a straightforward box diagram, familiar from \(K\bar{K}, D\bar{D},\) and \(B\bar{B}\) mixing. The commands for the labels are not shown here, they are discussed in section 2.5.6.

Let us start the diagram and pick up a thick pen:

\[
\begin{fmfgraph}(40,25)\\text{
ormalsize \begin{fmfpen} \text{thick} \end{fmfpen}}
\]

19
Table 6: Some of the available polygon shapes. Note that pull=1.0 is identical to pull=? for straight lines, but very different for smooth lines.

The incoming and outgoing vertices are placed on the left and right hand side, respectively:

\fml{i1,i2}
\fmr{o1,o2}

Now we tell \texttt{feynMF} how the arcs are connected.

\texttt{f fermion\{i1,v1,v3,o1\}}
\texttt{f fermion\{o2,v4,v2,i2\}}
\texttt{f photon\{v1,v2\}}
\texttt{f photon\{v3,v4\}}

Finally we tell \texttt{feynMF} to draw dots at the vertices and we're done.

\texttt{d \dotn\{v\}\{4\}}
\texttt{end\{fmfgraph\}}
With a little effort the layout of this diagram can actually be improved by enlarging the inner box, see page 29 below.

Here is the resonant $s$-channel contribution to $e^+e^- \to 4f$. (From now on, we do no longer display the

\[
\begin{fmfgraph}(40,25)
\fmfpen{thick}
\ldots
\end{fmfgraph}
\]

environment surrounding all pictures.)

\[
\begin{fmfgraph}(40,25)
\fmfleftn{i}{2}
\fmfrightn{o}{4}
\fmf{fermion}{i1,v1,i2}
\fmf{photon}{v1,v2}
\fmfblob{.15w}{v2}
\fmf{photon}{v2,v3}
\fmf{fermion}{o1,v3,o2}
\fmf{photon}{v2,v4}
\fmf{fermion}{o4,v4,o3}
\end{fmfgraph}
\]

And the resonant $t$-channel contribution:

\[
\begin{fmfgraph}(40,25)
\fmfleftn{i}{2}
\fmfrightn{o}{4}
\fmf{fermion}{i1,v1,v2,i2}
\fmf{photon}{v1,v3}
\fmf{fermion}{o1,v3,o2}
\fmf{photon}{v2,v4}
\fmf{fermion}{o4,v4,o3}
\end{fmfgraph}
\]

Two point loop diagrams pose another set of problems. We must have a way of specifying that one or more of the lines connecting the two vertices are not connected by a straight line. The options left, right and straight offer the possibility to connect two vertices by a semicircle detour, either on the left or on the right. Since by default all lines contribute to the tension between two vertices, the tension option allows us to reduce this tension. The next examples shows both options in action. The lower fermion line is given an tension of $1/3$ to make is symmetrical with the upper line with consists of three parts. The loop photon is using a detour on the right and does not contribute any tension.

\[
\begin{fmfgraph}(40,25)
\fmfleft{i1,i2}
\fmfright{o1}
\fmf{fermion, tension=1/3}{i1,v1}
\fmf{plain}{v1,v2}
\fmf{fermion}{v2,v3}
\fmf{photon,right, tension=0}{v2,v3}
\end{fmfgraph}
\]
The optional argument to `left` and `right` can be used to deform the corresponding contour as in the following example. The default value of `left` and `right` is 1.

Polygons are particularly useful for depicting non-perturbative contributions:

Scalar $\phi^6$-theory needs a little manual intervention to force the second on the opposite side:
2.5.6 Labels

Let us now come back to the examples on page 21 and discuss how to add the labels.

The macro

\fmflabel{<label>}{<v>}

is equivalent to

\fmfv{label=<label>}{<v>}

and adds the label \textit{<label>} to the vertex \textit{<v>}. In the current implementation, there can be only a single label for each vertex. Thus earlier calls to \texttt{\fmflabel} for the same vertex will be overwritten. \textit{<label>} will be placed with the \texttt{\put} command of the \LaTeX picture environment.\footnote{Note again that, as of version 1.03, it is no longer necessary to escape \LaTeX control sequences in arguments. Old files will continue to work, because \texttt{noexpand} is temporarily disabled.} Note that the \texttt{\fmfgraph*} environment must be used to use labels, they will silently disappear in \texttt{\fmfgraph}.

\texttt{\fmflabel} gives the user \textit{no} control on the placement of the the label (use the \texttt{\fmfv} macro for a more fine-grained control). The label is placed using the following algorithm:

1. The reference point of the box containing \textit{<label>} is placed at the distance \texttt{3thick} on the continuation of the straight line connecting the center of the picture with the vertex \textit{<v>}.

2. The reference point of the box is chosen such that the contents of the box is on the outside of the vertex (with respect to the center of the diagram). It is chosen from the four corners and the four midpoints of the sides.

Therefore the four external particles in the $B$-$\bar{B}$ mixing diagram on page 21 are labelled simply by:

\fmflabel{$\bar{b}$}{i1}
\fmflabel{$d$}{i2}
\fmflabel{$\bar{d}$}{o1}
\fmflabel{$b$}{o2}

Here is a more systematical demonstration of the default placement of labels:

\begin{figure}
\centering
\begin{fmfgraph*}(20,20)
\fmfsurroundn{v}{8}
\fmfdotn{v}{8}
\end{fmfgraph*}
\end{figure}
And here is a demonstration of the explicit placement of labels:\(^{19}\):

\[\text{\textbackslash fmfv}\{d.sh=circle,d.f=1,d.si=2\text{thin}\}\{c\}\]
\[\text{\textbackslash fmfv}\{l=-120,l.a=-120,l.d=2w\}\{c\}\]
\[\text{\textbackslash fmfv}\{l=-60,l.a=-60,l.d=.2w\}\{c\}\]
\[\text{\textbackslash fmfv}\{l=0,l.a=0,l.d=.2w\}\{c\}\]
\[\text{\textbackslash fmfv}\{l=60,l.a=60,l.d=.2w\}\{c\}\]
\[\text{\textbackslash fmfv}\{l=120,l.a=120,l.d=.2w\}\{c\}\]
\[\text{\textbackslash fmfv}\{l=180,l.a=180,l.d=.2w\}\{c\}\]

There is no equivalent to \texttt{\textbackslash fmflabel} for arcs. Here options to the \texttt{\textbackslash fmf} command have to be used. The default placement rules put the label on the outside at the midpoint of a curved arc. If the arc is straight, one should use the \texttt{label.side} option to push the label either to the \texttt{left} or to the \texttt{right}. This \texttt{label.dist} option is treated analogously to same option for vertices. Therefore the four internal particles in the $B\bar{B}$ mixing diagram on page 21 are labelled simply by adding options to the \texttt{\textbackslash fmf} commands:

\[\text{\textbackslash fmf}\{\text{fermion},\text{label}=$\bar{t}$,\,\bar{c},\,\bar{u}$,\}
\text{label.side=right}\}\{v1,v3\}\]
\[\text{\textbackslash fmf}\{\text{fermion},\text{label}=$t$,\,c,\,u$,\text{label.side=right}\}\{v4,v2\}\]
\[\text{\textbackslash fmf}\{\text{photon},\text{label}=$W^+$,\text{label.side=left}\}\{v1,v2\}\]
\[\text{\textbackslash fmf}\{\text{photon},\text{label}=$W^-$,\text{label.side=right}\}\{v3,v4\}\]

\subsection{2.5.7 Manipulating the layout}

The automatic layout algorithms of \texttt{feynMF} are rather simple, therefore it is sometimes necessary to allow for manual intervention from time to time.

\texttt{\textbackslash fmffreeze} Calculate the positions of the vertices based on the arcs which are defined up to this point. Usually this calculation is performed automatically at the end of the \texttt{fmfgraph} environment. Calling it explicitly is useful for later adding arcs that should not enter the calculation. The layout is chosen to minimize the overall length of all arcs. The length of each arc is weighted with the \texttt{tension} option, whose default value is 1. See section 2.5.8 for more information on \texttt{\textbackslash fmffreeze}.

\texttt{\textbackslash fmfforce} \texttt{\textbackslash fmfforce}\{\text{pos}\}\{\text{\textbackslash fmfforce}\}\{\text{\textbackslash \textbackslash fmfforce}v1,v3\}\} forces the position \{\text{pos}\} of the vertices \{\text{\textbackslash \textbackslash fmfforce}v1\}\,..., bypassing and overwriting the automatic layout. In all arguments that are METAFONT pairs (i.e. points), you can use the variable \texttt{w} and \texttt{h}, which are predefined to the width and the height of the whole diagram respectively. E.g. the center is specified as \{.5w,\ .5h\} and the lower right corner as \{w,\ 0\}. The center and the four corners of the current subgraph (see the \texttt{\textbackslash fmfsubgraph} environment on page 29) are available as \texttt{c,\ nw,\ ne,\ sw} and \texttt{se} (for \texttt{north-west} etc).

\texttt{\textbackslash fmffix} \texttt{\textbackslash fmffix}\{\text{dist}\}\{\text{\textbackslash fmffix}v1\}\,...\} shifts the position of the vertices \{\text{\textbackslash \textbackslash fmffix}v1\}\,... by \{\text{dist}\} from the automatic layout. This command is only useful after a \texttt{\textbackslash fmffreeze} of the corresponding vertex.

\texttt{\textbackslash fmffixed} \texttt{\textbackslash fmffixed}\{\text{dist}\}\{\text{\textbackslash fmffixed}v1\}\,...\} fixes the distance between subsequent vertices in the list \{\text{\textbackslash \textbackslash fmffixed}v1\}\,... to \{\text{dist}\}. This command should be used with care, because

\(^{19}\)Don’t be confused by the \texttt{\textbackslash fmfv} command. It is described below (see section 2.7.2) and takes the same arguments as the \texttt{\textbackslash fmfv} command. We use it here for convenience to place multiple vertices at the same point, i.e. the center.
it is possible to overconstrain the layout of the graph and the error messages
will be obscure for a novice user.

\[\text{\fmmfixedx{(dx)[\langle v1\rangle,\ldots]} is identical to \fmmfixed{{(\langle dx\rangle,\text{whatever})}\{\langle v1\}\text{,}\ldots\}} and \fmmfixedy{(dy)[\langle v1\rangle,\ldots]} is identical to \fmmfixed{{(\langle\text{whatever},\langle dy\rangle\rangle)\{\langle v1\}\text{,}\ldots\}}. These commands can be used to fix relative positions in one coordinate, while allowing movement in the other coordinate.}

\subsection{Skeletons}

The single most powerful concept for adjusting feynMF's layout decisions is the use of skeletons. By issuing a \texttt{\fmmfreeze} after specifying a subgraph (skeleton), we can fix the location of the skeleton as if the other arcs were not there. We can then successively add more subgraphs whose layout will be chosen with the skeleton remaining fixed. Similar effects can be achieved by giving some arcs a vanishing tension.

Consider the following example: suppose we want to draw a ladder diagram contributing to the quark form factor. Simply linking in the gluons does not produce a satisfactory result:

\begin{verbatim}
\fml{i1} \orm{o1,o2}
\mp{\text{photon}}{i1,v4}
\mp{\text{quark}}{o1,v1,v2,v3,v4,v5,v6,v7,o2}
\mp{\text{gluon}}{v1,v7}
\mp{\text{gluon}}{v2,v6}
\mp{\text{gluon}}{v3,v5}
\end{verbatim}

What went wrong? Obviously the gluons are bonding the quark lines too strongly. The fix is simple: just create a skeleton excluding the gluons

\begin{verbatim}
\fml{i1} \orm{o1,o2}
\mp{\text{photon}}{i1,v4}
\mp{\text{quark}}{o1,v1,v2,v3,v4,v5,v6,v7,o2}
\fmmfreeze
\mp{\text{gluon}}{v1,v7}
\mp{\text{gluon}}{v2,v6}
\mp{\text{gluon}}{v3,v5}
\end{verbatim}

and add the gluons later:

\begin{verbatim}
\fml{i1} \orm{o1,o2}
\mp{\text{photon}}{i1,v4}
\mp{\text{quark}}{o1,v1,v2,v3,v4,v5,v6,v7,o2}
\fmmfreeze
\mp{\text{gluon}}{v1,v7}
\mp{\text{gluon}}{v2,v6}
\mp{\text{gluon}}{v3,v5}
\end{verbatim}

Alternatively, we can use a vanishing tension, which will effectively exclude the gluons from the layout decisions:

\begin{verbatim}
\fml{i1} \orm{o1,o2}
\mp{\text{photon}}{i1,v4}
\end{verbatim}
Which method is more intuitive is largely a matter of taste and at the
discretion of the user.

The “crossed” version of this diagram is best drawn using the \rubout
option:

```
\fmfleft{i1} \fmfright{o1,o2}
\fmf{photon}{i1,v3}
\fmf{quark}{o1,v1,v2,v3,v4,v5,o2}
\fmffreeze
\fmf{gluon}{v1,v4}
\fmf{gluon,rubout}{v2,v5}
```

If \rubout is selected, it defaults to 2, but it can be changed to any
(reasonable) value > 1. The current implementation should work for
almost all user-defined line styles. However, it does not work correctly
for the predefined \double line styles yet.

Another instructive example is the following: imagine you want to draw
a typical non-resonant contribution to $e^+e^− \rightarrow 4f$. The obvious solution
doesn’t look right:

```
\fmfleft{i1,i2} \fmfright{o1,o2,o3,o4}
\fmf{fermion}{i1,v1,i2}
\fmf{boson}{v1,v2}
\fmf{fermion}{o1,v2,v3,o4}
\fmf{fermion}{o3,v4,o2}
```

Again, we should start with a skeleton

```
\fmfleft{i1,i2} \fmfright{o1,o2,o3,o4}
\fmf{fermion}{i1,v1,i2}
\fmf{boson}{v1,v2}
\fmf{fermion}{o1,v2,v3,o4}
```

and add the second $W$ later:

```
\fmfleft{i1,i2} \fmfright{o1,o2,o3,o4}
\fmf{fermion}{i1,v1,i2}
\fmf{boson}{v1,v2}
\fmf{fermion}{o1,v2,v3,o4}
```
Here’s another example that uses stretchable arcs. Diagrams of this kind are known as *rainbow* diagrams. If you’re using META O T and are watching this on a color device, you’ll see why.

```
\fmfpen{thick}
\fmfleft{i1,d1}
\fmfright{o1,d2}
\fmfn{plain}{i}\{4
\fmf{plain}{i4,v,o4}
\fmfn{plain}{o}\{4
\fmffreeze
\fmf{gluon,left,fore=red}{i1,o1}
\fmf{gluon,left,fore=green}{i2,o2}
\fmf{gluon,left,fore=blue}{i3,o3}
\fmfdotn{i}\{3
\fmfdotn{o}\{3
\fmfv{d.sh=circle,d.f=empty,d.si=.2w,b=(1,,0,,1),
1=\$\Sigma\$}{v}
```

Experience has shown that the method advocated in this section is more effective than fuzzing around with fractional *tension* parameters. Using \texttt{\fmfshift} or \texttt{\fmfforce} should be a last resort only.

### 2.5.9 Pulling strings

If you add to any arc one or more *phantom* arcs they will cause a tighter bonding between the vertices involved

\`
\fmf{fermion}{v1,v2}
\fmf{phantom}{v1,v2}
```

which is equivalent to

\`
\fmf{fermion,tension=2}{v1,v2}
```

The *phantom* arc has to be added before any \texttt{\fmffreeze} involving these vertices, of course.

Here is an example from deep inelastic scattering\textsuperscript{20}:

\`
\fmfleft{ip,il}
\fmfright{oq1,oq2,d1,oq3,d2,d3,ol}
\fmf{fermion}{ip,vp,vq,oq3}
```

\textsuperscript{20}Don’t be confused by the \texttt{\fmfi} command. It is described below (see section 2.7.1) and takes the same arguments as the \texttt{\fmfv} command. We use it here for adding to more lines parallel to the incoming proton line. They do not enter the layout decisions.
As it stands, all vertices come out too far to the right, because the greater number of outgoing lines pulls them over. Adding \texttt{phantom} makes the bond between the incoming vertices and the interactions tighter and produces a better balanced picture:

Equivalently, we could add 	exttt{tension} to the lines in question and we will get the same result:

Conversely, specifying a 	exttt{tension} < 1 will make the corresponding arcs more loose.
Reconsider the box graph on page 20 and reduce the tension on the inner lines:

\[ \text{This result is much nicer than the original.} \]

\section{Miscellaneous commands}

\subsection{Graphs in graphs}

The \texttt{fmfsubgraph} environment contains a subgraph, for which the galleries will be placed inside the rectangle of width \texttt{width} and height \texttt{height}, with lower left corner at \texttt{(x,y)}:

\begin{fmfsubgraph}(x,y)(width,height)

\end{fmfsubgraph}

The center and four corners are available as \texttt{c, nw, ne, sw and se} (for north-west etc). Because of the restrictions on the overall size of the diagram in METAFONT, this environment will, mainly be useful for preparing transparencies with \METAFONT.\O\,T.

Here is a not very serious application of this feature:

\begin{fmfgraph}(40,30)

\subgraphsample{a}

\end{fmfgraph}
More serious applications will use this feature to group together related diagrams and use other METAFONT drawing commands for graphical elements that are not available in the \LaTeX picture environment.

### 2.6.2 Reusing diagrams

\begin{fmfgraph}(20,15)
\fmf{dashes}{i,v,v,o}
\end{fmfgraph}
\begin{fmfgraph}(20,15)
\fmf{dashes}{i,v1,v2,o}
\end{fmfgraph}
\begin{fmfgraph}(20,15)
\fmf{fermion,left,tension=.3}{v1,v2,v1}
\end{fmfgraph}
\begin{fmfgraph}(20,15)
\fmfright{o} \fmf{dashes}{v1,v2,o}
\end{fmfgraph}
\begin{fmfgraph}(20,15)
\fmfleft{i}
\end{fmfgraph}

\begin{fmfgraph}(20,15)
\fmf{fermion,left,tension=.3}{v1,v2,v1}
\end{fmfgraph}
\begin{fmfgraph}(20,15)
\fmf{dashes}{i,v,v,o}
\end{fmfgraph}
\begin{fmfgraph}(20,15)
\fmf{dashes}{i,v1}
\end{fmfgraph}
\begin{fmfgraph}(20,15)
\fmfright{o} \fmf{dashes}{v1,v2,o}
\end{fmfgraph}
\begin{fmfgraph}(20,15)
\fmfleft{i}
\end{fmfgraph}

\begin{eqnarray}
\parbox{20mm}{\begin{fmfgraph}(20,15)
\fmfleft{i} \fmfright{o} \fmf{dashes}{i,v,v,o}
\end{fmfgraph}} & \propto & \Lambda^2 \\
\parbox{20mm}{\begin{fmfgraph}(20,15)
\fmfleft{i} \fmfright{o} \fmf{dashes}{i,v1} \fmf{dashes}{v2,o}
\end{fmfgraph}} & \propto & \Lambda^2 \\
\parbox{20mm}{\begin{fmfgraph}(20,15)
\fmf{fermion,left,tension=.3}{v1,v2,v1}
\end{fmfgraph}} & \propto & \ln \Lambda^2
\end{eqnarray}

The result is:

\begin{align}
\begin{array}{c}
\begin{fmfgraph}(20,15)
\fmf{dashes}{i,v,v,o}
\end{fmfgraph}
\end{array}
\propto \Lambda^2 & \quad (1) \\
\begin{array}{c}
\begin{fmfgraph}(20,15)
\fmfleft{i} \fmfright{o} \fmf{dashes}{i,v1}
\end{fmfgraph}
\end{array}
\propto \Lambda^2 & \quad (2) \\
\begin{array}{c}
\begin{fmfgraph}(20,15)
\fmf{dashes}{i,v,v,o}
\end{fmfgraph}
\end{array} + \begin{array}{c}
\begin{fmfgraph}(20,15)
\fmfleft{i} \fmfright{o} \fmf{dashes}{i,v1}
\end{fmfgraph}
\end{array}
\propto \ln \Lambda^2 & \quad (3)
\end{align}

### 2.6.3 Grouping

\texttt{fmfgroup} Make the enclosed \texttt{fmfset} assignments local to this environment.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>thin</td>
<td>1pt</td>
<td>thin arcs</td>
</tr>
<tr>
<td>thick</td>
<td>1.5thin</td>
<td>thicker arcs</td>
</tr>
<tr>
<td>arrow_len</td>
<td>4mm</td>
<td>length of arrow head</td>
</tr>
<tr>
<td>arrow_ang</td>
<td>15</td>
<td>opening angle of arrow head</td>
</tr>
<tr>
<td>curly_len</td>
<td>3mm</td>
<td>length of one curl</td>
</tr>
<tr>
<td>dash_len</td>
<td>3mm</td>
<td>length of one dash</td>
</tr>
<tr>
<td>dot_len</td>
<td>2mm</td>
<td>distance of two dots</td>
</tr>
<tr>
<td>wiggly_len</td>
<td>4mm</td>
<td>length of one wiggles</td>
</tr>
<tr>
<td>wiggly_slope</td>
<td>60</td>
<td>inclination of wiggles</td>
</tr>
<tr>
<td>zigzag_len</td>
<td>2mm</td>
<td>length of a zig-zag period</td>
</tr>
<tr>
<td>zigzag_width</td>
<td>2thick</td>
<td>width of zig-zag lines</td>
</tr>
<tr>
<td>decor_size</td>
<td>5mm</td>
<td>default size of vertex decors</td>
</tr>
<tr>
<td>dot_size</td>
<td>4thick</td>
<td>diameter of dots</td>
</tr>
</tbody>
</table>

Table 7: Available style parameters.

### 2.6.4 Changing parameters

\texttt{\fmfset} This command can be used to change the parameters in table 7 as follows:

\texttt{\fmfset\{parameter\}\{value\}}

Note that these parameters are not stored in the graph data structure for the individual vertices and arcs. Instead the current values at the time of \texttt{\fmfdraw} are used.

### 2.6.5 Shrinking

\texttt{\fmfshrink} Shrink the linewidths and similar parameters in the enclosed section.

### 2.6.6 Debugging

\texttt{\fmftrace} \texttt{\fmfnotrace} Enable and disable tracing of the layout decisions. This is not necessarily printed in an intuitive format, but can be helpful for debugging.

\texttt{\fmfdisplay} Enable online displays. \texttt{\fmfstopdisplay} will halt METAPOST everytime a graph is complete.

### 2.6.7 Multiple vertices and arcs

\texttt{\fmffor} The environment

\texttt{\begin{fmffor}\{(var)\}\{(from)\}\{(step)\}\{(to)\}\{body\}}

\texttt{\end{fmffor}}

executes \texttt{(body)} multiple times, setting \texttt{(var)} to \texttt{(from)}, \texttt{(from)+(step)}, \ldots, \texttt{(to)}.

An application of this above \texttt{feynMF} feature is shown in figure 7, which is generated by calling the \texttt{\LaTeX} macro

\texttt{\def\EulerHeisenberg#1{}}

31
Figure 7: Higher order terms in the Euler-Heisenberg lagrangian.

\begin{fmfgraph}(40,25)
  \fmfpen{thick}
  \fmfsurroundn{e}{#1}
  \begin{fmffor}{n}{1}{1}{#1}
    \fmf{photon}{e[n],i[n]}
  \end{fmffor}
  \fmfcyclen{fermion,tension=#1/8}{i}{#1}
\end{fmfgraph}

with the arguments 4, 6, 8, and 10, respectively.

Similarly, we can draw the diagrams from many particle physics in figures 8 and 9:

\def\PPRing#1{%
  \begin{fmfgraph}(20,20)
    \fmfsurroundn{v}{#1}
    \fmfdotn{v}{#1}
    \fmfcyclen{fermion,right=0.25}{v}{#1}
    \fmfcyclen{fermion,left=0.25}{v}{#1}
  \end{fmfgraph}
}\def\PHRing#1{%
  \begin{fmfgraph}(20,20)
    \fmfsurroundn{v}{#1}
    \fmfdotn{v}{#1}
    \fmfcyclen{fermion,right=0.25}{v}{#1}
    \fmfrccyclen{fermion,right=0.25}{v}{#1}
  \end{fmfgraph}}

2.7 Immediate mode

In addition to the automatic layout of vertices, \texttt{feynMF} features an immediate mode, in which \texttt{feynMF}'s drawing commands operate directly on \texttt{METAFONT}'s
pairs and paths. You might want to consult The \textsc{Metafont} Book [4] or the \textsc{Metafont} manual [5] for further information on the available path expressions.

2.7.1 Arcs

\texttt{\textbackslash fmfi} Immediate mode’s brother of \texttt{\textbackslash fmf}.

\begin{verbatim}
\fmfi{(style)[,(opt)]=[val],...]}{(p)}
\end{verbatim}

draws a line of style \textlangle style\textrangle on path \textlangle p\textrangle. Use the \texttt{vpath} function in \textlangle p\textrangle (after \texttt{\textbackslash fmffreeze}) to access the \textsc{metafont} path connecting two vertices: \texttt{vpath\{\textlangle tag\textrangle\}\{\textlangle from\textrangle,\textlangle to\textrangle\}}. The optional numeric \textlangle tag\textrangle can be used together with a matching \texttt{tag} option to \texttt{\textbackslash fmfi} to disambiguate arcs that connect the same vertices. You have to prepend each name of a vertex in \texttt{vpath}’s arguments with two underscores (e.g. \texttt{v1} becomes \texttt{\_\_v1}). This is necessary for avoiding name-clashes with some reserved words in \textsc{metafont} (sparks in DEK’s terminology).

2.7.2 Vertices

\texttt{\textbackslash fmfv} Immediate mode’s brother of \texttt{\textbackslash fmfv}.

\begin{verbatim}
\fmfv\{\textlangle shape\rangle=[val],[,(opt)]=[val],...]\{\textlangle v\textrangle\}
\end{verbatim}

draws a vertex at position \textlangle v\textrangle. Note that here \textlangle v\textrangle is a \textsc{metafont} pair and \textit{not} a \texttt{feyn\textsc{mf}} vertex name. The former’s equivalent of the latter can be accessed (after \texttt{\textbackslash fmffreeze}) with the \texttt{vloc} function: \texttt{vloc\{\textlangle vertex\textrangle\}}. Again, you have to prepend the name of the vertex in \texttt{vloc}’s argument with two underscores (e.g. \texttt{v1} becomes \texttt{\_\_v1}).

2.7.3 Declarations

\texttt{\textbackslash fmipath} The argument(s) are declared \textsc{metafont} paths.

\texttt{\textbackslash fmipair} The argument(s) are declared \textsc{metafont} pairs (coordinates).
2.7.4 Assignments

\texttt{\textbf{\textbackslash fmfequ}} Establish equality for the two arguments, i.e. \texttt{\textbackslash fmfequ\{lval\}\{rval\}} translates to \texttt{lval=rval}.

\texttt{\textbf{\textbackslash fmfsiset}} Assign the second argument to the first, i.e. \texttt{\textbackslash fmfsiset\{lval\}\{rval\}} translates to \texttt{lval:=rval}.

Specifying equality of two variables is a very different operation from assignment in METAFONT. See The METAFONT Book [4] for details on METAFONT’s builtin equation solver.

2.7.5 Examples

Here is a non-trivial example of immediate mode, which shows some useful tricks. The non-trivial aspect of the diagram in question is that it has lines broken in two, denoting particles coupling to a condensate.

We start the diagram with a skeleton (the phantom lines are shown as thin lines for clarity):

\begin{verbatim}
\fmfleft{i}
\fmfright{o}
\fmf{dots}{i,v1}
\fmf{dots}{v2,o}
\fmf{phantom,left,tension=0.2,tag=1}{v1,v2}
\fmf{phantom,left,tension=0.2,tag=2}{v2,v1}
\fmfdot{v1,v2}
\fmfposition
\end{verbatim}

Add the fermions to the skeleton

\begin{verbatim}
\fmfipath{p[]}
\fmfiset{p1}{vpath1(__v1,__v2)}
\fmfiset{p2}{vpath2(__v2,__v1)}
\fmfi{fermion}{subpath (0,length(p1)/3) of p1}
\fmfi{fermion}{subpath (2length(p1)/3,length(p1)) of p1}
\fmfi{fermion}{subpath (0,length(p2)/3) of p2}
\fmfi{fermion}{subpath (2length(p2)/3,length(p2)) of p2}
\end{verbatim}

Add condensates and a gluon

\begin{verbatim}
\def\cond#1#2{%
    \fmfiv{d.sh=cross,d.ang=#1,d.siz=5thick}{#2}}
\cond{30}{point length(p1)/3 of p1}
\cond{-30}{point 2length(p1)/3 of p1}
\cond{30}{point length(p2)/3 of p2}
\cond{-30}{point 2length(p2)/3 of p2}
\fmfi{gluon}{point length(p1)/10 of p1}
    -- point 11length(p1)/12 of p1}
\end{verbatim}

\begin{verbatim}
\def\vert#1{%
    \fmfiv{d.sh=circle,d.f=1,d.siz=2thick}{#1}}
\vert{point length(p1)/12 of p1}
\vert{point 11length(p1)/12 of p1}
\end{verbatim}
Figure 10: Circular gluons.

Here’s an interesting abuse of \texttt{feynMF} (see the next section for \texttt{\fmfcmd}):

\begin{verbatim}
\begin{fmfgraph*}(40,40)
\fmfipair{o,xm,xp,ym,yp}
\fmfiequ{o}{(.5w,.1h)}
\fmfiequ{xm}{(0,.1h)}
\fmfiequ{xp}{(w,.1h)}
\fmfiequ{ym}{(.5w,0)}
\fmfiequ{yp}{(.5w,h)}
\fmfiv{l=$x$,l.a=-135,l.d=2mm}{xp}
\fmfiv{l=$y=x^2$,l.a=-135,l.d=2mm}{yp}
\fmfcmd{draw xp--xm; draw yp--ym;}
\fmfcmd{draw (o + (-xs,ys)) for n = -9 upto 10: --(o + (xs*(n/10),ys*((n/10)**2))) endfor;}
\end{fmfgraph*}
\end{verbatim}

Finally, for the curious, here is how to draw the circular gluons in figure 10:

\begin{verbatim}
\fmf{gluon}{fullcircle scaled .5w shifted (.5w,.5h)}
\fmf{gluon}{reverse fullcircle scaled .5w shifted (.5w,.5h)}
\end{verbatim}

\section*{2.8 Raw METAFONT}

Some more advanced features of \texttt{feynMF} are more conveniently accessed through raw \texttt{METAFONT} commands. This can either be achieved by preparing a \texttt{METAFONT} input file or by using \texttt{\fmfcmd} extensively. The latter approach is usually more convenient.

The \texttt{\fmfcmd} macro writes its argument into the \texttt{METAFONT} input file generated by \texttt{feynMF}. While some experience in using \texttt{METAFONT} doesn’t hurt here, this
approach can simplify the production of complex diagrams considerably. Note that no semicolon is appended, the user has to provide it explicitly.

2.8.1 Extending \texttt{feynMF}

A prominent example for using raw \texttt{METAFONT} is provided by the option to add new styles for arcs. There is of course always one more style that \texttt{must} be added to the default list. But increasing this list without bounds will eventually slow down \texttt{feynMF} and increase its memory requirements. It is therefore better to allow users to define their own styles. This is done with the \texttt{METAFONT} macro \texttt{style_def}, which defines a macro that will be called to do the drawing and registers this macro with \texttt{feynMF} so that it can be used in the first argument to \texttt{\fmf}.

The macro takes one argument of type \texttt{path} and is responsible for drawing the arc on this path. If \texttt{META O T}'s color functionality is to be used, the color aware functions \texttt{cdraw}, \texttt{cfill}, \texttt{cfilldraw}, \texttt{ccutdraw} and \texttt{cdrawdot} should be used instead of \texttt{draw}, etc.

After the following \texttt{style_def}, a new style \texttt{crossed} will be available:

\begin{verbatim}
\FMFCmd{\
  vardef cross_bar (expr p, len, ang) =
    ((-len/2,0)--(len/2,0))
    rotated (ang + angle direction length(p)/2 of p)
    shifted point length(p)/2 of p
  enddef;
  style_def crossed expr p =
    cdraw p;
    ccutdraw cross_bar (p, 5mm, 45);
    ccutdraw cross_bar (p, 5mm, -45)
  enddef;}
\end{verbatim}

it can be used just like any other style:

\begin{verbatim}
\fmfleft{i}
\fmfright{o}
\fmf{plain}{i,v,o}
\fmf{crossed}{v,v}
\fmfdot{v}
\end{verbatim}

And here is an (esthetically questionable!) attempt at wiggly lines with arrows:

\begin{verbatim}
\FMFCmd{\
  style_def wiggly_arrow expr p =
    cdraw (wiggly p);
    shrink (2);
    cfill (arrow p);
  endshrink;
  enddef;}
\end{verbatim}
Note how the `shrink` macro (which is the `METAFONT` equivalent of the `fmfshrink` environment) is used to temporarily double the dimensions of the arrowhead which is constructed by the `arrow` macro.

In particular theorists beyond the standard model are likely to need a lot of different line styles. The aficionados of majorana neutrinos might find the following two useful:

\fmcmd{%
    style_def majorana expr p =
        cdraw p;
        cfill (harrow (reverse p, .5));
        cfill (harrow (p, .5))
    enddef;
    style_def alt_majorana expr p =
        cdraw p;
        cfill (tarrow (reverse p, .55));
        cfill (tarrow (p, .55))
    enddef;}

Note the use of the `harrow` and `tarrow` functions which return an arrowhead on the given fraction of the path, with reference points at the head (`harrow`) or tail (`tarrow`). The `arrow` function used above is equivalent to `marrow(p,.5)`, which has the reference point at the center of the arrowhead. Having the three different reference point available is important for supporting arcs of substantially differing lengths.

A problem of the current implementation is that the endpoints of double lines don’t match smoothly at vertices:

\fmpen{thick}\fml{<2}\fmr{<2}
\fm{shaded}{z}{4}
\fmd{12,z3}
\fmd{11,z4}
\fmd{z1,r1}
\fmd{z2,r2}
\fmf{dbl_plain_arrow}{l2,z3}
\fmf{dbl_plain_arrow}{l1,z4}
\fmf{dbl_plain_arrow}{z1,r1}
\fmf{dbl_plain_arrow}{z2,r2}
\fmfreeze\fmfdraw
\fmd{d.siz=2thick,d.sh=circle}{z}{4}

One way around is to add dots of the right size at the vertices after the arcs have been drawn.
Without the `\fmmfreeze\fmdraw`, the arcs would “know” about the dots and would be shortened.

A more elegant solution is to define line styles with dots at the head,

```latex
\fmcmd{vardef endpoint_dot expr p =
  save oldpen; pen oldpen;
  oldpen := currentpen;
  pickup oldpen scaled 3;
  cdrawdot p;
  pickup oldpen;
  enddef;}
\fmcmd{style_def hd_double expr p =
  draw_double p;
  endpoint_dot point infinity of p;
  enddef;}
\fmcmd{style_def hd_dbl_plain_arrow expr p =
  draw_hd_double p;
  shrink (1.5);
  cfill (arrow p);
  endshrink;
  enddef;}

```

or both

```latex
\fmcmd{style_def htd_double expr p =
  draw_double p;
  endpoint_dot point 0 of p;
  endpoint_dot point infinity of p;
  enddef;}
\fmcmd{style_def htd_dbl_plain_arrow expr p =
  draw_htd_double p;
  shrink (1.5);
  cfill (arrow p);
  endshrink;
  enddef;}
```

```latex
38
```
which can be used as follows to give an equivalent result:

\fmfpen{thick}\fmfleftn{l}{2}\fmfrightn{r}{2}  
\fmfpolyn{shaded}{z}{4}  
\fmf{hd_dbl_plain_arrow}{l2,z3}  
\fmf{hd_dbl_plain_arrow}{l1,z4}  
\fmf{td_dbl_plain_arrow}{z1,r1}  
\fmf{td_dbl_plain_arrow}{z2,r2}

2.9 Common traps, trouble shooting and frequently asked questions (FAQs)

2.9.1 Value is too large

The following will be one of the most frequent errors:

! Value is too large (4097).
\texttt{<recently read> ;

\begin{verbatim}
beginchar->...rdp:=(EXPR3);w:=hround(charwd*hppp);
h:=...
1.685 ...char(64, 40*2.84526pt#, 25*2.84526pt#, 0)
\end{verbatim}

It can have one of two causes:

- **METAFONT** was invoked without specifying the **mode**. This case is easily fixed by looking up the correct METAFONT mode in the file **modes.mf** that comes with the METAFONT distribution. This mode must be specified on the command line as \texttt{mode:=laserjet} for HP Laserjets at 300dpi, as \texttt{mode:=ljfour} for HP Laserjets at 600dpi, as \texttt{mode:=nexthi} for NeXT laser printers at 400dpi, etc, just to name three of the more common laser printers in the physics community.

- The diagrams are too large for the printer at hand. This case is actually not very likely, because even at 1200dpi the diagrams can be as large as 86mm. For the popular laserprinter resolution of 300dpi, even 346mm are possible. Last time I checked, the diagrams for this manual could be generated for a Linotype Linotronic 300 at 2540dpi (\texttt{mode:=linosuper}), but failed in the standard **proof** mode at 2601.72dpi. In fact, in the current **modes.mf** file, the Chelgraph IBX at 9600dpi and the Alphatype CRS at 5333 + 1/3dpi are the only typesetters that can not be used to typeset this manual.\footnote{It would be trivial to shrink the diagrams by 1% to make them work in **proof** mode (accidentally, the largest diagram is 40mm wide, while 4096/2601.72dpi corresponds to 39.99mm). However, I prefer METAFONT to give an error message if the user forgot to specify the mode. It is much more obscure when METAFONT works without errors but the dvi driver fails to find the generated bitmap file.}

\footnote{If someone wants to use **feynMF** with one of these high end typesetters, I would be glad to try to help them out with kludges.}
2.9.2 Diagrams in the document are never updated

There are two known reasons why diagrams may not be updated after the source file has been changed:

- Some dvi file previewers (e.g. xdvi(1) under UNIX) do not reread font information if the tfm or pk files have changed, even though they reread the dvi file if it has changed. Therefore you have to restart such previewers if you have made changes in diagrams to see these changes on the screen.

- Some dvi drivers (e.g. dvips(1) under UNIX) do not work with the gf files directly, but convert them with an external program to pk format first. On later occasions, the dvi driver will then use the pk file which is out of date with respect to the sources and the gf file. The only known fix is to delete the pk files before running the dvi driver.

2.9.3 Diagrams show up in the wrong spot

If you are using feynMF with L\TeX\s \texttt{\includeonly} feature, you should watch out for the following situation:

\begin{verbatim}
\includeonly{bar1,bar3}
\begin{fmffile}{foograph}
\include{bar1}
\include{bar2}
\include{bar3}
\end{fmffile}
\end{verbatim}

where bar1.tex defines graph #1, bar2.tex graph #2 and bar3.tex defines graph #3. If you now proceeded to add graphs to bar1.tex, you will notice that a second graph is accepted, but instead of the new third graph, the old graph #3 appears. What happens is that L\TeX\ stores the value of the counter for fmffgraphs in each .aux file so that because bar2.tex is not processed, this counter is always reset to 3 at the beginning of bar3.tex.

Even though this situation appears to be contrived, it actually occurred in real life applications and the resulting error is very confusing.

The only “fix” for this problem would be to use a private counter behind L\TeX\s back. Unfortunately, it appears that this will violate the \textit{principle of minimal surprise} even more. It is therefore usually a good idea to reprocess the complete document when the number of graphs has changed in an \texttt{\include} file. The other solution is to have a separate fmffgraph environment for each \texttt{\include} file.

2.9.4 Spurious labels show up

If spurious labels show up in your diagrams, this is most likely caused by old label files (e.g. foo.t\langle n\rangle) still lying around. Just delete these files and rerun \TeX\ and METAFONT (or META \O T respectively).
2.10 Known bugs

2.10.1 Chaotic manual

This is being worked on. It should probably be rewritten from scratch, but I don’t have enough time at the moment (this is a spare time activity).

2.10.2 Delayed error messages

This can’t be fixed. The problem is that errors can manifest themselves only a long time after the corresponding source line has been read. Since \TeX doesn’t allow to access the current source line number, there is no way to store this information along with the other information on the graph. I can only hope to have enough sanity checks in place some day that error messages from META\-FONT won’t occur.

2.10.3 Multiple tadpoles

Currently, feynMF will not layout multiple tadpoles at a single vertex automatically. This could be fixed in principle, but these fixes would cause other problems which are more inconvenient than having to lay out tadpoles manually.

2.10.4 Hard limits

Currently the most severe limitation lies in the size of the generated pictures. The largest number META\-FONT can represent internally is 4095.99998 and this is also the largest value any coordinate measured in pixels can assume. At the most popular laserprinter resolution of 300 dots per inch (dpi), this corresponds to a horizontal and vertical extension of about 346mm, which is plenty and we’re more likely to hit the internal limits on the complexity of a picture. However, at the proof mode resolution of 2601.72dpi, this is reduced to slightly less than 40mm and we’re running the risk of arithmetic overflow in internal calculations much earlier.

There are two potential solutions of different scope and complexity:

- Since John Hobby’s META\-OT is now available without a non-disclosure agreement from AT&T, one solution is to replace META\-FONT by META\-OT, which doesn’t suffer from the size limitations. This comes with a small price paid in reduced portability of the generated output, but as already stated above in the case of axodraw, the ubiquity of PostScript printers (and the free GhostScript interpreter) makes this a minor point.

- The more ambitious solutions would be virtual graphs, i.e. graphs which are larger than the current limit enforced by numeric overflow at higher resolutions. This could be implemented by calculating the layout of a miniature graph and afterwards distributing the full graph among several META\-FONT characters.

Acknowledgements

I am most grateful to Wolfgang Kilian, who pushed feynMF’s predecessor feynman.mf to its limits [12]. Discussions with him triggered a lot of good
ideas. Thanks also to my students and the people on *The Net* for suggestions, portability fixes and for volunteering as guinea pigs.

**References**


**Distribution**

*\texttt{feynMF} is available by anonymous internet ftp from any of the Comprehensive \texttt{TeX} Archive Network (CTAN) hosts

\begin{verbatim}
ftp.shsu.edu, ftp.tex.ac.uk, ttp.dante.de
\end{verbatim}

in the directory

\begin{verbatim}
macros/latex/contrib/supported/feynmf
\end{verbatim}

It is also available from the host

\begin{verbatim}
crunch.ikp.physik.th-darmstadt.de
\end{verbatim}

in the directory
Unsupported snapshots of my work in progress are provided as

```
pub/ohl/feynmf.versions/feynmf-current.tar.gz
```

There are two mailing lists

```
feynmf-announce@crunch.ikp.physik.th-darmstadt.de
feynmf-bugs@crunch.ikp.physik.th-darmstadt.de
```

open for subscription. The former should carry only important announcements,
of new versions in particular. To subscribe, send mail to the (electronic) mailing
list manager

```
majordomo@crunch.ikp.physik.th-darmstadt.de
```

and not to the lists itself. The following commands (on a line in the body of
the mail, not in the subject) are useful:

```
subscribe feynmf-announce
unsubscribe feynmf-announce
help
```

Index

Numbers written in italic refer to the page where the corresponding entry is
described, the ones underlined to the code line of the definition, the rest to the
code lines where the entry is used.

- T, --notfm, 13
c, --clean, 13
f, --force, 13
h, --help, 12
m mode, --mode mode, 13
n, --noexec, 13
q, --quiet, 13
t, --tfm tfm, 13
v, --version, 12
arcs=arcs, 14, 16, 33
blobs=blobs, 17
color=color, 19
crossed arcs=crossed arcs, 36
defining new styles=defining new styles, 36
dots=dots, 17
dots=..., 17
dots=env, 17
dots=env, 17
dots=env, 17

```
fmfgraph* ...
```

```
fmfgraph* ...
```

```
fmfgraph* ...
```

```
fmfgraph* ...
```

```
fmfgraph* ...
```

43
\texttt{fmfgroup} = \texttt{fmfgroup} \texttt{fmfshift} = \texttt{fmfshift}\texttt{fmfshrink} = \texttt{fmfshrink}\texttt{fmfi} = \texttt{fmfi} \texttt{fmfsequ} = \texttt{fmfsequ}\texttt{fmfipair} = \texttt{fmfipair}\texttt{fmfipath} = \texttt{fmfipath}\texttt{fmfiset} = \texttt{fmfiset}\texttt{fmfiv} = \texttt{fmfiv}\texttt{fmfkeep} = \texttt{fmfkeep}\texttt{fmflabel} = \texttt{fmflabel}\texttt{fmfleft} = \texttt{fmfleft}\texttt{fmfleftn} = \texttt{fmfleftn}\texttt{fmfn} = \texttt{fmfn}\texttt{fmfnotrace} = \texttt{fmfnotrace}\texttt{fmfpen} = \texttt{fmfpen}\texttt{fmfpoly} = \texttt{fmfpoly}\texttt{fmfpolyn} = \texttt{fmfpolyn}\texttt{fmfrcyclen} = \texttt{fmfrcyclen}\texttt{fmfreuse} = \texttt{fmfreuse}\texttt{fmfright} = \texttt{fmfright}\texttt{fmfrightn} = \texttt{fmfrightn}\texttt{fmfrpoly} = \texttt{fmfrpoly}\texttt{fmfset} = \texttt{fmfset}\texttt{fmfshift} = \texttt{fmfshift}\texttt{fmfstopdisplay} = \texttt{fmfstopdisplay}\texttt{fmfstraight} = \texttt{fmfstraight}\texttt{fmfsubgraph} = \texttt{fmfsubgraph}\texttt{fmfsurround} = \texttt{fmfsurround}\texttt{fmfsurroundn} = \texttt{fmfsurroundn}\texttt{fmftop} = \texttt{fmftop}\texttt{fmftopn} = \texttt{fmftopn}\texttt{fmftrace} = \texttt{fmftrace}\texttt{fmfv} = \texttt{fmfv}\texttt{fmfvn} = \texttt{fmfvn}\texttt{galleries} = \texttt{galleries}\texttt{internal vertices} = \texttt{internal vertices}\texttt{labels} = \texttt{labels}\texttt{looping} = \texttt{looping}\texttt{polygons} = \texttt{polygons}\texttt{OPTIONS} = \texttt{OPTIONS}\texttt{parameters} = \texttt{parameters}\texttt{polygons} = \texttt{polygons}\texttt{styles, defining} = \texttt{styles, defining new}\texttt{subgraphs} = \texttt{subgraphs}\texttt{tadpoles} = \texttt{tadpoles}\texttt{tracing} = \texttt{tracing}\texttt{vertices, external} = \texttt{vertices, external}\texttt{vertices, internal} = \texttt{vertices, internal}\texttt{vertices} = \texttt{vertices}\texttt{online displays} = \texttt{online displays}\texttt{shapes, defining new} = \texttt{shapes, defining new}