feynMF: Drawing Feynman Diagrams with LaTeX and METAFONT

Thorsten Ohl[*]

Technische Hochschule Darmstadt
Schloßgartenstr. 9
D-64289 Darmstadt
Germany

March 5, 1995

Abstract

feynMF is a LaTeX package for easy drawing of professional quality Feynman diagrams with METAFONT. feynMF lays out most diagrams satisfactorily from the structure of the graph without any need for manual intervention. Nevertheless all the power of METAFONT is available for the most obscure cases.

Copying

Contents

1 Introduction **2**
1.1 Purpose and scope 2
1.2 Relation to similar packages 3
1.2.1 Feynman 3
1.2.2 Axodraw 3

[*]e-mail: `Thorsten.Ohl@Physik.TH-Darmstadt.de`

1.2.3 Mfpic . . . 3
1.3 Historical note . . . 4
1.4 Projects . . . 4
1.4.1 Documented METAFONT Interface . . . 4
1.4.2 Less Formal Graphs . . . 4
1.4.3 Virtual Graphs . . . 5
1.5 Conclusion . . . 5

2 Usage **5**
2.1 Invocation . . . 5
2.2 Basic usage . . . 6
2.3 Examples . . . 12
2.4 Labels . . . 14
2.5 Advanced usage . . . 15
2.5.1 Shrinking . . . 15
2.5.2 Multiple vertices and arcs . . . 15
2.5.3 Tight arcs . . . 16
2.5.4 Loose arcs . . . 17
2.5.5 Avoiding continuously tight and loose arcs . . . 18
2.5.6 Raw METAFONT . . . 19
2.6 Limitations . . . 20

1 Introduction

1.1 Purpose and scope

In recent years, TEX[1] [1] and LATEX[2] [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], which provide line drawings like the one in figure 1. As an de facto standard for the inclusion of more complex graphics has emerged the inclusion of PostScript [3] 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

[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.

reproduced from a very concise specification, if the software is able to choose a reasonable layout (semi-)automatically.
METAFONT[4] [4] and METAPOST[5] [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] was conceived.
`feynMF` supports METAFONT as well as METAPOST (under the names `feynmp.sty` and `feynmp.mp`).

1.2 Relation to similar packages

Before we start, a couple of words about some complementary packages on the market are in order. First of all: I failed to do my homework and didn't try hard enough to find [6] in a library. I'm sure that in there is a smarter way of returning information from METAFONT to TEX. Those who don't know the literature are doomed to reinvent the wheel. But this isn't a scholarly work and reinventing the wheel was *fun!*

1.2.1 Feynman

Micheal Levine's `feynman` package [7] is implemented on top of the standard LATEX `picture` environment [2]. This makes it completely portable (no need for a correct METAFONT installation), but it requires manual layout and the graphics output is (though very useful) less than perfect.

1.2.2 Axodraw

Jos Vermaseren's `axodraw` package [8] uses `\special`s to access PostScript primitives for drawing diagrams. This approach is inherently not portable (though the ubiquity of PostScript printers makes this a minor point) but at least as flexible as the present one. It still requires manual layout, though.

1.2.3 Mfpic

Last, but not least, I have to mention Thomas Leathrum's `mfpic` [9], which was the main inspiration for moving `feynMF`'s user interface from METAFONT to

[4] METAFONT is a trademark of Addison Wesley Publishing Company.
[5] John Hobby's METAPOST is a modified version of METAFONT which generates (encapsulated) PostScript output. METAPOST 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 a trademark of anybody.

TEX. It might not have been unreasonable to design and implement `feynMF` as a package on top of `mfpic`[7], But closer inspection shows that `feynMF` and `mfpic` are fairly orthogonal. `mfpic` is ideally suited for handling simple graphical building blocks in formally unconstrained contexts, a task which will be accomplished preferably by direct manual placement of objects. `feynMF` on the other hand works in the highly formal context of Feynman diagrams (or any other labeled graphs for that matter), which can be draw automatically from a *specification.*
It will of course be tempting to add more features from `mfpic`'s realm to `feynMF` in the future, but I will try to be conservative in that respect.

1.3 Historical note

This code has a rather long history[8]. Most of the sections **??**, **??**, **??**, and **??** started in 1989 as `feynman.mf`, a library of METAFONT macros for drawing Feynman diagrams in my thesis. The layout had to be chossen completely manually, which required a long edit-process-preview cycle and made `feynman.mf` awkward to use. Nevertheless, it survived for five years without major modifications, only slight enhancements had been made. Early in 1994, I became aware of Thomas Leathrum's `mfpic` [9]. This inspired me to shift the user interface from METAFONT to LATEX, because this allows a smoother blending of the LATEX `picture` environment with `feynMF`. While doing this and after having been taught by Tim Stelzer's and Bill Long's `MADGRAPH` [10] that minimizing the length of the graph gives surprisingly good results for tree graphs[9], I added the graph manipulation code in section **??**.
And, of course, `feynMF` is biased towards those kinds of graphs that my students, my colleagues and myself have to draw most often. Comments for improvements from other areas of physics are welcome.

1.4 Projects

1.4.1 Documented METAFONT Interface

In a future version, `feynMF` will have a more convenient and supported set of macros at the METAFONT level, including more general drawing primitives.

1.4.2 Less Formal Graphs

In addition to the well defined Feynman graphs of perturbation theory there are also less formal and more illustrative graphs in use. For example the "quark diagrams" familiar from papers on hadronic weak decays involve curved lines illustrating the formation of bound states, etc. Future versions of `feynMF` will provide better support for such graphs, without the need for raw METAFONT coding.

[7]In fact a change in this direction would be rather easy.

[8]Which is a partial explanation, if not excuse, for its slightly incoherent structure.

[9]I had thought about this earlier myself, but foolishly discarded the idea. I didn't expect such a "too simple" method to give esthetically pleasing results for loop graphs, which were my main concern.

1.4.3 Virtual Graphs

In a future version, `feynMF` will be able to draw "virtual" graphs, i.e. graphs which are larger than the current limit enforced by numeric overflow at higher resolutions. This can be implemented by calculating the layout of a miniature graph and afterwards distributing the full graph among several METAFONT characters.

1.5 Conclusion

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

2 Usage

2.1 Invocation

Instructing LaTeX to use `feynMF` is as simple as[10]

```
\usepackage{feynmf}
```

and calling METAFONT should not be harder as

```
mf '\mode:=localmode; input foobar'
```

where `foobar` is the name of your METAFONT output file and `localmode` is your local printer (e.g. `laserjet`). The hard part usually lies in instructing TeX and your favorite `dvi`-driver how to find the generated `tfm` and `gf` (resp. `pk`) files. This is highly system dependent and can be trivial (as in the standard UNIX[11] TeX installations) or almost impossible (as under MVS). Please consult your local guide or local "TeX Wizards" on this point.
If you have METAPOST, then you can use it by placing

```
\usepackage[dvips]{feynmp}
```

in your LaTeX source. Here `dvips` is an option for the LaTeX `egraphics` package which is used for including the generated PostScript files. It should be replaced by your local supported `dvi` driver. Calling METAPOST is usually even simpler

```
mp foobar
```

[10] As given, this applies to LaTeX. But the installation file `feynmf209.ins` allows to generate special versions `feynmf209.sty` and `feynmp209.sty` which are compatible with the obsolete LaTeX version 2.09. These files are to be used as documentstyle options.

[11] UNIX was a trademark of UNIX Systems Laboratory, but is rumored to have been donored to X/Open.

since there is no mode to be picked.
Please keep in mind that `feynMF` has been developed for LaTeX 2ε 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 LaTeX2e, which is the one and onloy supported LaTeX right now.
The flow of information depicted in figure 1 looks much more complicated than it is. The important feature is that there a two sets of files which can be used to distribute a document:

1. Iff the recipient has a working METAFONT installation (which shouldn't be a problem, except for some impoverished commercial implementations), the document can be typset from the LaTeX source *alone*, by running LaTeX, METAFONT and LaTeX again (the latter step might have to be repeated to get cross references right).

2. Another possibility (which doesn't require METAFONT on the recipient's side), is to distribute the LaTeX source, the `tfm` and `gf` files (or `pk` files respectively) along with the label files with extension `t`n (where n as an integer). Distributing the METAFONT `log` files is a possible alternative for the latter, but discouraged, because these are prone to be erased accidentally.

I should add one caveat here: 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. Thus you have to restart such previewers if you have made changes in diagrams to see these changes on the screen.

2.2 Basic usage

The basic features of `feynMF` are (or rather "should be") available through the LaTeX interface. Not knowledge of METAFONT is necessary.

`fmffile` Upto 255 characters can be placed into one METAFONT file, they are enclosed in a single `fmffile` environment. The environment takes the filename as argument. Currently `feynMF` does *not* check that the 255 character limit per file is not overrun.

```
\begin{fmffile}{foobar}
  ...
\end{fmffile}
```

Note that the filename for the METAFONT file given in the argument of the `fmffile` environment *must not* be identical to the LaTeX source file name, because the METAFONT `.log` would be overwritten and LaTeX can no longer access the information in this `.log` file.

`fmfchar` Draw one character and place it here. Arguments are `(`⟨*width*⟩`,`⟨*height*⟩`)` as in:

```
\begin{fmfchar}(40,25)
  ...
\end{fmfchar}
```

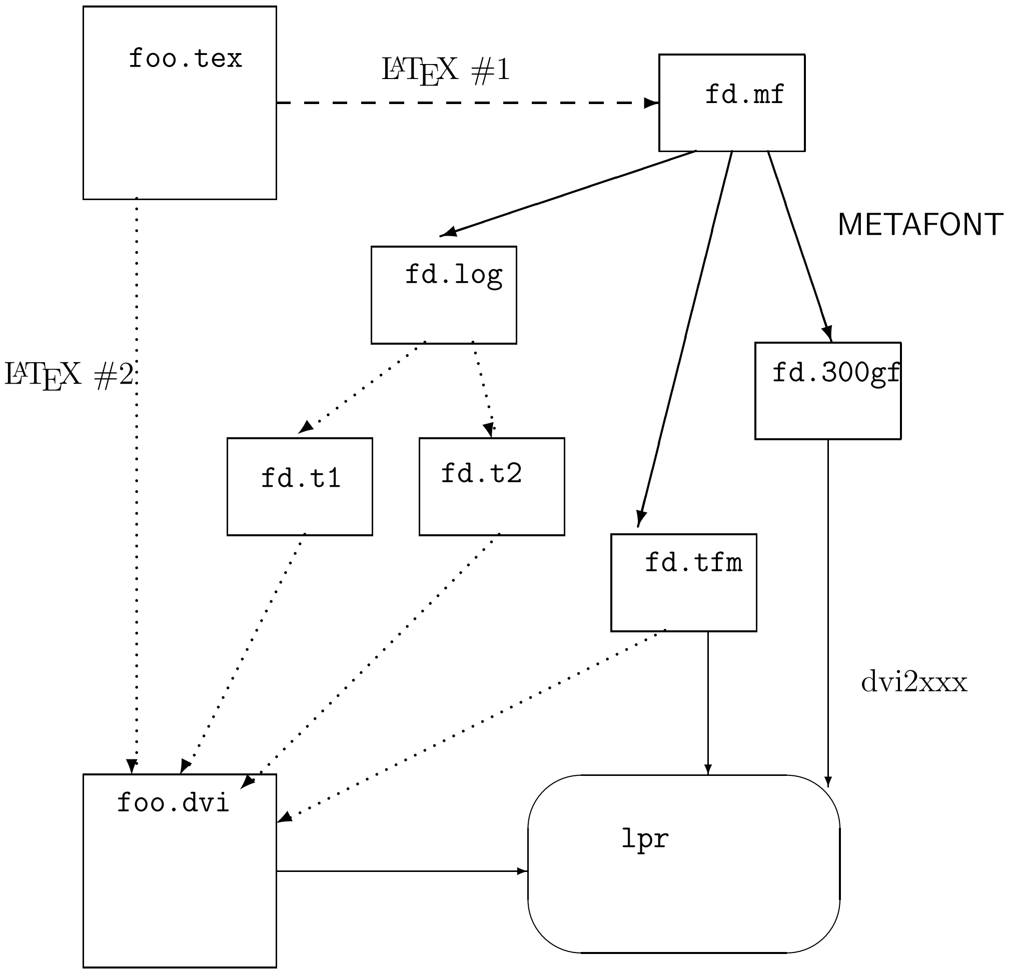


Figure 1: Flow of information in a feynMF application: The first LaTeX pass is shown with a dashed line. The METAFONT pass is shown with the full lines and the second LaTeX pass with dotted lines. The final dvi translation step is shown with thin lines.

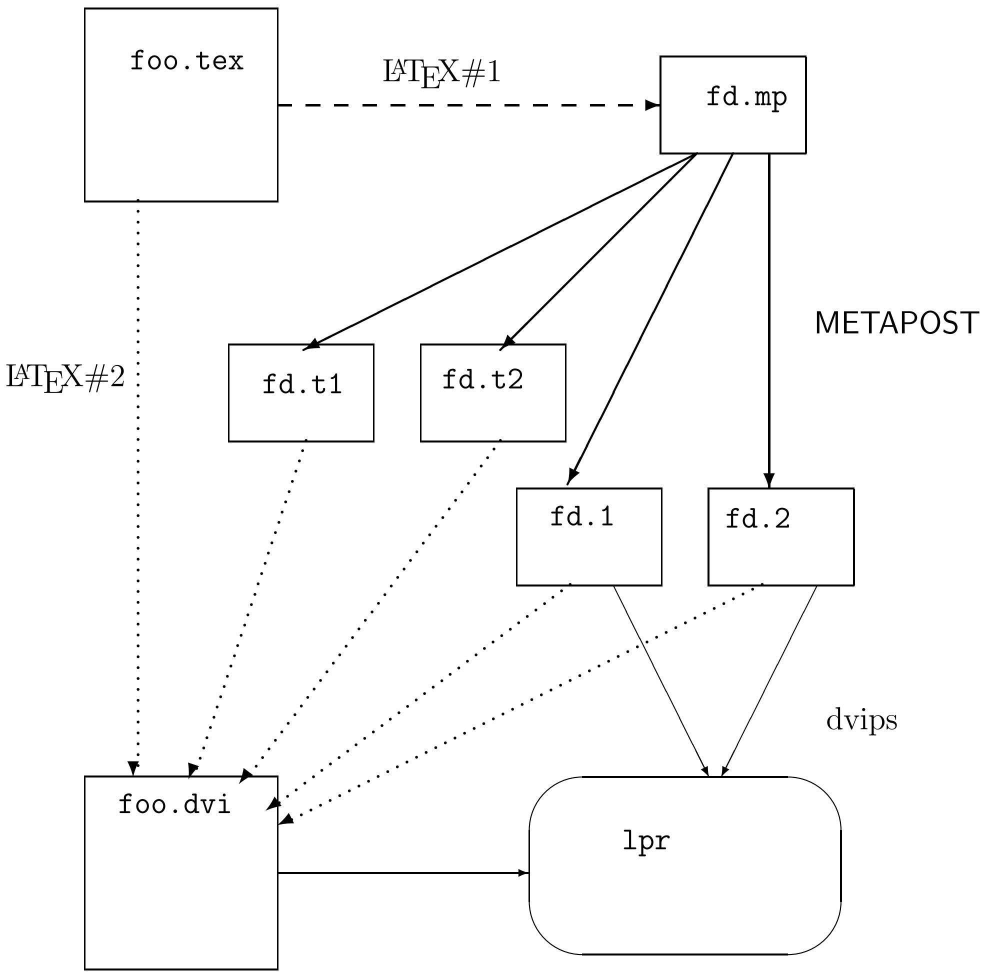


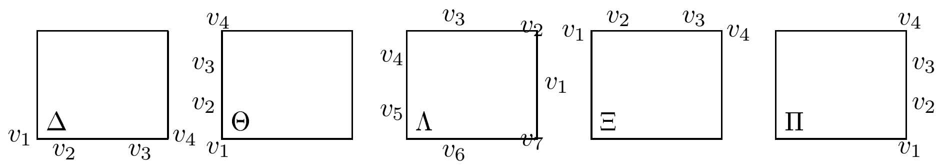
Figure 2: Flow of information in a `feyn`MF application if METAPOST is used instead of METAFONT.

This environment does *not* support labels, use `fmfchar*` if your diagrams contains labels.

`fmfchar*` Same as `fmfchar`, but enclosed in a `picture` environment of the same size and supporting LaTeX labels.

```
\begin{fmfchar*}(40,25)
  ...
\end{fmfchar*}
```

`\fmfleft` `\fmfright` `\fmfbottom` `\fmftop` `\fmfsurround` Positioning of external vertices has to done explicitely. The technical reason is that they would otherwise collapse with their neighbors, but practical reasons also suggest to give the user full control here. `\fmfleft{`⟨*v1*⟩`[,...]}` places the vertices in the comma separated list ⟨*v1*⟩,... equidistantly on a smooth path on the left side of the diagram. `\fmfright{`⟨*v1*⟩`[,...]}` does the same thing on the right. Similarly `\fmfbottom` and `\fmftop`, while `\fmfsurround{`⟨*v1*⟩`[,...]}` places its arguments on smooth path surrounding the diagram:



`\fmfpen` Pick up a pen of the specified size. `\fmfpen{`⟨*weight*⟩`}` is used for changing the weight (i.e. thickness) of the lines. Predefined sizes are `thin` and `thick`.

`\fmf` This is the the most frequently used macro in `feynMF` applications. `\fmf{`⟨*style*⟩ `[,`⟨*opt*⟩`[=`⟨*val*⟩`],...]}{`⟨*v1*⟩`,`⟨*v2*⟩`[,...]}` connects the vertices *v1,v2,...* with a line of style ⟨*style*⟩, using a set of options ⟨*opt*⟩ with (optional) value ⟨*val*⟩. 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 `dashes`, `dots`, `phantom` and `plain` styles can optionally be decorated with an arrow as shown above. All styles, including `curly` and `wiggly`, can be doubled. But arrows are not available for the latter two, because esthetically pleasing results can not be expected. The `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. The supported options are collected in table 2[12]. 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.

`\fmfv` Declare vertices with options: `\fmfv{`⟨*shape*⟩`[=`⟨*val*⟩`][,`⟨*opt*⟩`[=`⟨*val*⟩`],...]}{`⟨*v1*⟩ `[,...]}` 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 mechanism as above is in effect. The available `shape`s are listed in various filling styles in table 4[13].

`\fmfblob` Draw a blob of the specified diameter at the vertices. `\fmfblob{`⟨*diameter*⟩`}{`⟨*v1*⟩

12

One particulary useful further option would be `smooth`, allowing for several lines joined smoothly. Early experimentation has shown however, that the results are not always

Name	Example	Parameters	Aliases
curly	Σ	curly_len	gluon
dbl_curly	Υ	curly_len	
dashes	Φ	dash_len	
dashes_arrow	Ψ	dash_len	scalar
dbl_dashes	Ω	dash_len	
dbl_dashes_arrow	ff	dash_len	
dots	fi	dot_len	
dots_arrow	fl	dot_len	ghost
dbl_dots	ffi	dot_len	
dbl_dots_arrow	ffl	dot_len	
phantom	ı		
phantom_arrow	ȷ		
plain	`		vanilla
plain_arrow	´		fermion, electron, quark
dbl_plain	ˇ		double
dbl_plain_arrow	˘		double_arrow, heavy
wiggly	¯	wiggly_len	boson, photon
dbl_wiggly	˚	wiggly_len	

Table 1: Available line styles

Name	Explanation
tension	draw a tighter (> 1) or more loos (< 1) arc
left	draw on a halfcircle on the left
right	draw on a halfcircle on the right
straight	draw on a straight line
label	TEX text for labeling the arc
label.side	force placement of the label on the left or right
label.dist	place label at a distance dist
label.pos	relative position of the label (not implemented yet!)
decoration.shape	shape of decoration (not implemented yet!)
decoration.size	size of decoration (not implemented yet!)
decoration.pos	relative position of decoration (not implemented yet!)
decoration.filled	fill, shade or hatch decoration (not implemented yet!)
decoration.angle	rotate decoration relative to default (not implemented yet!)

Table 2: Available line options

Name	Explanation
`label`	TEX text for labeling the vertex
`label.angle`	force placement of the label at the given angle from the vertex
`label.dist`	place label at a distance `dist`
`decoration.shape`	shape of decoration
`decoration.size`	size of decoration
`decoration.filled`	fill, shade or hatch decoration
`decoration.angle`	rotate decoration

Table 3: Available vertex options

	`filled=-.5`	`filled=0`	`filled=.5`	`filled=1`
`circle`	¸	ß	æ	œ
`square`	ø	Æ	Œ	Ø
`triangle`	-	!	”	#
`diamond`	$	%	&	’
`pentagon`	(	)	*	+
`hexagon`	,	-	.	/
`triagram`	0	1	2	3
`tetragram`	4	5	6	7
`pentagram`	8	9	:	;
`hexagram`	¡	=	¿	?

Table 4: Available vertex shapes and fill styles.

`[,...]}` is equivalent to `\fmfv{decor.shape=circle,decor.filled=.5,decor.size=`⟨*diameter*⟩`}{`⟨*v1*⟩`[,...]}`

`\fmfdot` Draw a dot at the vertices given as arguments. `\fmfdot{`⟨*v1*⟩`[,...]}` is equivalent to `\fmfv{decor.shape=circle,decor.filled=1,decor.size=2thick}{`⟨*v1*⟩`[,...]}`

`\fmfposition` 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 `fmfchar` environment. Calling it explicitely, is useful for adding arcs which should not enter the calculation.

`\fmfforce` `\fmfforce{`⟨*pos*⟩`}{`⟨*v1*⟩`[,...]}` forces the position ⟨*pos*⟩ of the vertices ⟨*v1*⟩..., bypassing the automatic layout.

`\fmfshift` `\fmfforce{`⟨*dist*⟩`}{`⟨*v1*⟩`[,...]}` shifts the position of the vertices ⟨*v1*⟩... by ⟨*dist*⟩ from the automatic layout.

`\fmffixed` `\fmfforce{`⟨*dist*⟩`}{`⟨*v1*⟩`[,...]}` fixes the distance between subsequent vertices in the list ⟨*v1*⟩... to ⟨*dist*⟩. This command should be used with care, because it is possible to overconstrain the layout of the graph and the error messages will be obscure to a novice user.

2.3 Examples

As an example, consider drawing a straightforward box diagram, familier from K-$\bar{K}$, D-$\bar{D}$, and B-$\bar{B}$ mixing. The commands for the labels are not shown here,

what one expects and that there is a lot of room for abuse.

[13]If the variable `feymfwizard` is `true` (e.g. after calling the `fmfwizard` macro), it is also possible to specify any METAFONT expression that evaluates to a `path`. Naturally, this has to used with great care, because strange errors can be triggered by typos!

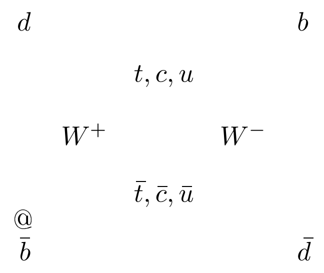
they are discussed in section 2.4

We start the diagram and pick up a thick pen:

```
\begin{fmfchar}(40,25)
  \fmfpen{thick}
```

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

```
    \fmfleft{i1,i2}
    \fmfright{o1,o2}
```



Now we tell `feynMF` how the arcs are connected.

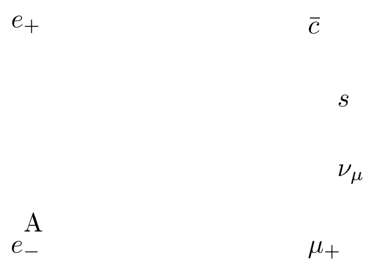
```
    \fmf{fermion}{i1,v1,v3,o1}
    \fmf{fermion}{o2,v4,v2,i2}
    \fmf{photon}{v1,v2}
    \fmf{photon}{v3,v4}
```

Finally we tell `feynMF` to draw dots at the vertices and we're done.

```
    \fmfdot{v1,v2,v3,v4}
  \end{fmfchar}
```

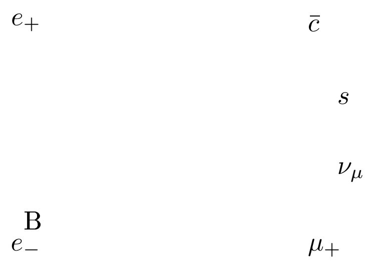
Here is the resonant s-channel contribution to $e^+e^- \to 4f$. (From now on, we do no longer display the `\begin{fmfchar}(40,25)`, `\fmfpen{thick}`, …, `\end{fmfchar}` surrounding all pictures.)

```
    \fmfleft{i1,i2}
    \fmfright{o1,o2,o3,o4}
      \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}
```



And the resonant t-channel contribution:

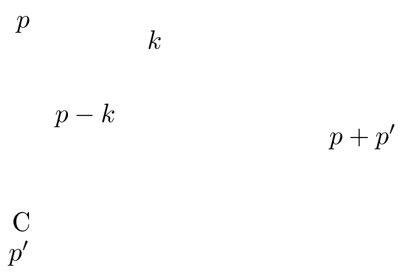
```
    \fmfleft{i1,i2}
    \fmfright{o1,o2,o3,o4}
    \fmf{fermion}{i1,v1,v2,i2}
      \fmf{photon}{v1,v3}
        \fmf{fermion}{o1,v3,o2}
      \fmf{photon}{v2,v4}
        \fmf{fermion}{o4,v4,o3}
```



Actually, these three diagrams can be improved slightly by using `phantom` arcs,

which will be discussed in the next section.
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.

```
\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}
\fmf{plain}{v3,i2}
\fmf{photon}{v1,o1}
```



2.4 Labels

Let us now come back to the examples on page 13 and discuss how to add the labels.

`\fmflabel` The macro `\fmflabel{`⟨*label*⟩`}{`⟨*v*⟩`}` adds the label ⟨*label*⟩ to the vertex ⟨*v*⟩. In the current implementation, there can be only a single label for each vertex. Thus earlier calls to `\fmflabel` for the same vertex will be overwritten. ⟨*label*⟩ will be placed with the `\put` command of the LaTeX `picture` environment. *It is absolutely necessary to quote* each *TeX control sequence appearing in* ⟨*label*⟩ *with* `\noexpand`. *Otherwise all kinds of disasters are bound to happen, causing at the very least some obscure error messages!* Note that the `fmfchar*` environment must be used to use labels, they will silently disappear in `fmfchar`.
`\fmflabel` gives the user *no* control on the placement of the the label (see below for a more fine-grained control provided by the options to the `\fmfv` macro). The label is placed using the following algorithm:

1. The reference point of the box containing ⟨*label*⟩ is placed at the distance `3thick` on the continuation of the straight line connecting the center of the picture with the vertex ⟨*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 13 are labelled simply by:

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

Explain more of the `label` option and the default placement rules.

2.5 Advanced usage

2.5.1 Shrinking

`fmfshrink` Shrink the lineswidths and similar parameters in the enclosed section.

2.5.2 Multiple vertices and arcs

`\fmfleftn` `\fmfrightn` `\fmfbottomn` `\fmftopn` `\fmfsurroundn` `\fmfvn` `\fmfdotn` `\fmfblobn` `fmffor` `\fmfcyclen` `\fmfrcyclen`

The macro is`\fmfleftn` is similar to `\fmfleft`, but `\fmfleftn{`$\langle v\rangle$`}{`$\langle n\rangle$`}` places the vertices $\langle v[1]\rangle \ldots \langle v[n]\rangle$. Analogously for the macros `\fmfrightn`, `\fmfbottomn`, `\fmftopn` and `\fmfsurroundn`.
The macro `\fmfvn` is similar to `\fmfv`, but `\fmfvn{`$\langle vertex\rangle$`}{`$\langle v\rangle$`}{`$\langle n\rangle$`}` places the vertices $\langle v[1]\rangle \ldots \langle v[n]\rangle$.
The macros `\fmfdotn` and `\fmfblobn` are similar to the `\fmfdot` and `\fmfblob`, but `\fmfdotn{`$\langle v\rangle$`}{`$\langle n\rangle$`}` places the vertices $\langle v[1]\rangle \ldots \langle v[n]\rangle$. Analogously for `\fmfblobn`.
The environment `\begin{fmffor}{`$\langle var\rangle$`}{`$\langle from\rangle$`}{`$\langle step\rangle$`}{`$\langle to\rangle$`}`$\langle body\rangle$`\end{fmffor}` executes $\langle body\rangle$ multiple times, setting $\langle var\rangle$ to $\langle from\rangle$, $\langle from\rangle + \langle step\rangle, \ldots, \langle to\rangle$.
The macro `\fmfcyclen{`$\langle style\rangle$`}{`$\langle v\rangle$`}{`$\langle n\rangle$`}` cyclically connects the vertices $\langle v[1]\rangle \ldots \langle v[n]\rangle$. `\fmfcyclen` operates in reverse order.
An advanced application of the above `feynMF` features is shown in figure 3, which is generated by calling the TEX macro

```
\def\EulerHeisenberg#1{%
   \begin{fmfchar}(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{fmfchar}}
```

with the arguments 4, 6, 8, and 10, respectively.
Similarly, we can draw the diagrams in figures 4 and 4 from many particle physics:

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

D E

F G

Figure 3: Higher order terms in the Euler-Heisenberg lagrangian.

H I J K

Figure 4: Particle-particle ring diagrams

2.5.3 Tight arcs

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 `\fmfposition` involving these vertices, of course. Here is an example from deep inelastic scattering. (We do not show the `\fmfcmd{}`s in this example which are used for decorating the

L M N O

Figure 5: Particle-hole ring diagrams

incoming proton and do not affect feynMF's layout decisions.)

```
\fmfleft{ip,il}
\fmfright{oq1,oq2,d1,oq3,d2,d3,ol}
\fmf{fermion}{ip,vp,vq,oq3}
\fmf{fermion}{vp,oq1}
\fmf{fermion}{vp,oq2}
\fmf{photon}{vl,vq}
\fmf{fermion}{il,vl,ol}
\fmfblob{.15w}{vp}
\fmfdot{vq}
```

P

As it stands, all vertices come out too far to the right, because the greater number of outgoing lines pulls them over. Adding `\fmf{phantom}` makes the bond between the incoming vertices and the interactions tighter and produces a better balanced picture:

```
\fmfleft{ip,il}
\fmfright{oq1,oq2,d1,oq3,d2,d3,ol}
\fmf{fermion}{ip,vp,vq,oq3}
\fmf{phantom}{ip,vp}
\fmf{fermion}{vp,oq1}
\fmf{fermion}{vp,oq2}
\fmf{photon}{vl,vq}
\fmf{fermion}{il,vl,ol}
\fmf{phantom}{il,vl}
\fmfblob{.15w}{vp}
\fmfdot{vq}
```

Q

2.5.4 Loose arcs

Adding arcs *after* any `\fmfposition` involving these vertices will make these arcs loose, i.e. they will not contribute to the bonding between the vertices. 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:

```
\fmfleft{i1} \fmfright{o1,o2}
\fmf{photon}{i1,v4}
\fmf{quark}{o1,v1,v2,v3,v4,v5,v6,v7,o2}
\fmf{gluon}{v1,v7}
\fmf{gluon}{v2,v6}
\fmf{gluon}{v3,v5}
```

R

What went wrong? Obviously the gluons are bonding the quark lines too strongly. The fix is simple: just exclude the gluons from the calculation and add them later as infinitely stretchable:

```
\fmfleft{i1} \fmfright{o1,o2}
\fmf{photon}{i1,v4}
\fmf{quark}{o1,v1,v2,v3,v4,v5,v6,v7,o2}
\fmfposition
\fmf{gluon}{v1,v7}
\fmf{gluon}{v2,v6}
\fmf{gluon}{v3,v5}
```

S

Another instructive example is the following: imagine you want to draw a typical non-resonant contribution to $e^+e^- \to 4f$. The obvious solution doesn's look right.

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

T

One way to fix it is to `\fmfshift` the three rightmost vertices by hand:

```
\fmfleft{i1,i2}
\fmfright{o1,o2,o3,o4}
\fmf{fermion}{i1,v1,i2}
\fmf{photon}{v1,v2}
\fmf{fermion}{o1,v2,v3,o4}
\fmf{photon}{v3,v4}
\fmf{fermion}{o3,v4,o2}
\fmfposition
\fmfshift{-.1w,0}{v2,v3,v4}
```

U

A smarter solution is again to make certain arcs stretchable:

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

V

2.5.5 Avoiding continuously tight and loose arcs

I have been very reluctant to implement continuously tight and loose arcs in `feynMF`, because it introduces to much opportunity for “fiddling” on the user’s part. However, since the present implementation blends rather nicely with the options syntax, I have decided to add it anyway. I hope that most diagrams will be created without too much “fiddling”.

Figure 6: Circular gluons.

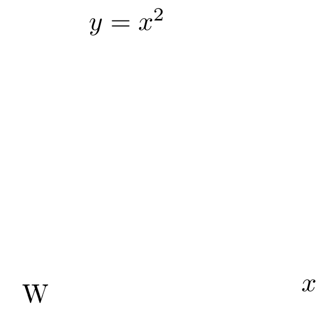
2.5.6 Raw METAFONT

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

`\fmfcmd` The `\fmfcmd` macro writes its argument into the METAFONT input file generated by feynMF. While some experience in using METAFONT doesn't hurt here, this approach can simplify the production of complex diagrams considerably.

Here's an interesting "abuse" of feynMF. Using the undocumented METAFONT macros (like `vdraw_vertex_label`) is possible, though not supported. In a future versions, these features will be made available through supported interfaces.

```
\begin{fmfchar*}(40,40)\fmfcmd{
  pair o,xm,xp,ym,yp;
  pair v.loc; string v.lbl;
  o:=(.5w,.1h);
  xm:=(0,.1h); xp:=(w,.1h);
  ym:=(.5w,0); yp:=(.5w,h);
  v.loc := xp; v.lbl := "$x$";
  v.lbl.ang := -135; v.lbl.dist := 2;
  vdraw_vertex_label v;
  v.loc := yp; v.lbl := "$y=x^2$";
  vdraw_vertex_label v;
  pickup pencircle scaled thin;
  draw xm--xp; draw ym--yp;
  pickup pencircle scaled thick;
  xs:=xpart(xp-o); ys:=ypart(yp-o);
  draw (o + (-xs,ys)) for n = -9 upto 10:
    --(o + (xs*(n/10),ys*((n/10)**2)))
  endfor;}
\end{fmfchar*}
```



Finally, for the curious, here is how to draw the circular gluons in figure 6:

```
\fmfcmd{draw_gluon (fullcircle scaled .5w shifted (.5w,.5h));}
\fmfcmd{draw_gluon (reverse fullcircle scaled .5w shifted (.5w,.5h));}
```

2.6 Limitations

Currently the most severe limitation lies in the size of the generated pictures. The largest number METAFONT 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 METAPOST is now available without a non-disclosure agreement from AT&T, one solution is to replace METAFONT by METAPOST, which doesn't suffer from the size limitations[14] 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 will be "virtual pictures" (see section 1.4.3).

Acknowledgements

I am most grateful to Wolfgang Kilian, who pushed `feynMF`'s predecessor `feynman.mf` to its limits [11]. His needs were valuable input to `feynMF`. Thanks also to Angelika Himmler and Uwe Mayer for acting as guinea pigs.

References

[1] Donald E. Knuth, *The TEXbook*, Addison-Wesley, Reading, 1986.

[2] Leslie Lambort, *LATEX — A Documentation Preparation System*, Addison-Wesley, Reading, 1985.

[3] Michel Goosens, Frank Mittelbach, and Alexander Samarin, *The LATEX Companion*, Addison-Wesley, Reading, 1994.

[4] Donald E. Knuth, *The METAFONTbook*, Addison-Wesley, Reading, 1986.

[5] John D. Hobby, *A User's Manual for METAPOST*, Computer Science Report #162, AT&T Bell Laboratories, April 1992.

[6] Alan Hoenig, *When TEX and METAFONT Work Together*, in *Proceedings of the 7th European TEX Conference, Prague*, p. 1, September 1992.

[7] Micheal J. S. Levine, Comp. Phys. Comm. **58** (1990) 181.

[8] Jos Vermaseren, `axodraw`.

[14]Right now, `feynMF`'s METAPOST support is still somewhat kludged, but the functionality is there.

[9] Thomas E. Leathrum, `mfpic`.

[10] Tim Stelzer and Bill Long, `MADGRAPH`, hep-ph/93mmxxx.

[11] Wolfgang Kilian, Doctoral Thesis, Technical University Darmstadt, 1994.

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

environments:¿`fmfchar*`, *9*
environments:¿`fmfchar`, *6*
environments:¿`fmffile`, *6*
environments:¿`fmffor`, *15*
environments:¿`fmfshrink`, *15*

fmf= `\subitem *+\fmf+, \usage{9}`
fmfblob= `\subitem *+\fmfblob+, \usage{9}`
fmfblobn= `\subitem *+\fmfblobn+, \usage{15}`
fmfbottom= `\subitem *+\fmfbottom+, \usage{9}`
fmfbottomn= `\subitem *+\fmfbottomn+, \usage{15}`
fmfchar*=`fmfchar*` (environment), *9*
fmfchar=`fmfchar` (environment), *6*
fmfcmd= `\subitem *+\fmfcmd+, \usage{19}`
fmfcyclen= `\subitem *+\fmfcyclen+, \usage{15}`
fmfdot= `\subitem *+\fmfdot+, \usage{12}`
fmfdotn= `\subitem *+\fmfdotn+, \usage{15}`
fmffile=`fmffile` (environment), *6*
fmffixed= `\subitem *+\fmffixed+, \usage{12}`
fmffor=`fmffor` (environment), *15*
fmfforce= `\subitem *+\fmfforce+, \usage{12}`
fmflabel= `\subitem *+\fmflabel+, \usage{14}`
fmfleft= `\subitem *+\fmfleft+, \usage{9}`
fmfleftn= `\subitem *+\fmfleftn+, \usage{15}`
fmfpen= `\subitem *+\fmfpen+, \usage{9}`
fmfposition= `\subitem *+\fmfposition+, \usage{12`
fmfrcyclen= `\subitem *+\fmfrcyclen+, \usage{15}`
fmfright= `\subitem *+\fmfright+, \usage{9}`
fmfrightn= `\subitem *+\fmfrightn+, \usage{15}`
fmfshift= `\subitem *+\fmfshift+, \usage{12}`
fmfshrink=`fmfshrink` (environment), *15*
fmfsurround= `\subitem *+\fmfsurround+, \usage{9`
fmfsurroundn= `\subitem *+\fmfsurroundn+, \usage`
fmftop= `\subitem *+\fmftop+, \usage{9}`
fmftopn= `\subitem *+\fmftopn+, \usage{15}`
fmfv= `\subitem *+\fmfv+, \usage{9}`
fmfvn= `\subitem *+\fmfvn+, \usage{15}`